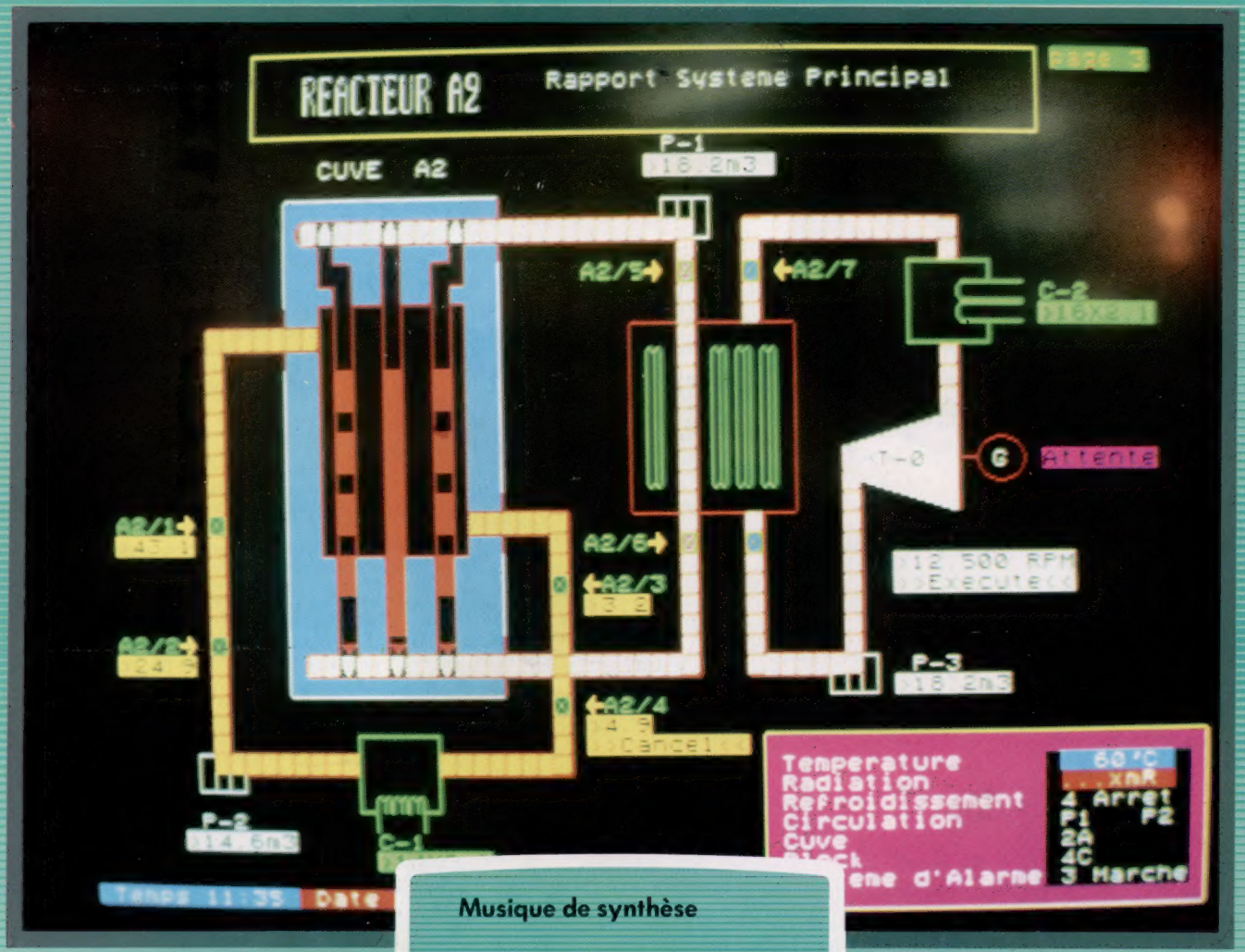
ABC INFORMATIQUE

N°53

COURS D'INFORMATIQUE PRATIQUE ET FAMILIALE



- Musique de synthèse
- Bibliothèque de programmes
- L'Apple IIc
- Construire un interrupteur

EDITIONS ATLAS

M6062-53-12-F 85FB-3,80FS-$1.95

Page manquante
(publicité et colophon)

Musique de synthèse

Parmi tous les systèmes musicaux construits autour d'un micro-ordinateur, MIDI offre la gamme d'applications la plus large, une introduction dans l'univers de la musique électronique.

Il est souvent difficile d'apprendre à lire la musique, même lorsqu'on est un assez bon exécutant. Maîtriser des éléments comme l'indication de la mesure, les dièses et les bémols peut être très long et très laborieux. Il est rarement facile d'établir une relation immédiate entre les indications écrites et ce qui est entendu.

Le problème provient surtout de la nature de la musique elle-même : pour être significative, elle doit être formée d'une série d'événements prenant place dans le temps. Si le débutant ne peut arriver à suivre la notation visuelle qu'en arrêtant régulièrement la musique en question, les indicateurs de durée de la partition deviennent inutiles. De même, il peut perdre beaucoup de temps à essayer d'interpréter une séquence particulière alors que la musique continue.

MIDI élimine tous ces obstacles. Il permet de stocker un enregistrement de synthétiseur dans une mémoire de micro-ordinateur et, avec le logiciel approprié, donne un affichage graphique de la musique jouée. Cela signifie que si un *do* est joué au clavier du synthétiseur, ce *do* sera affiché à l'écran sur une portée à cinq lignes. Si un accord en *si* mineur est maintenu pendant un certain temps, les composants harmoniques de l'accord — *si, ré* et *fa* — seront affichés ensemble avec la durée appropriée.

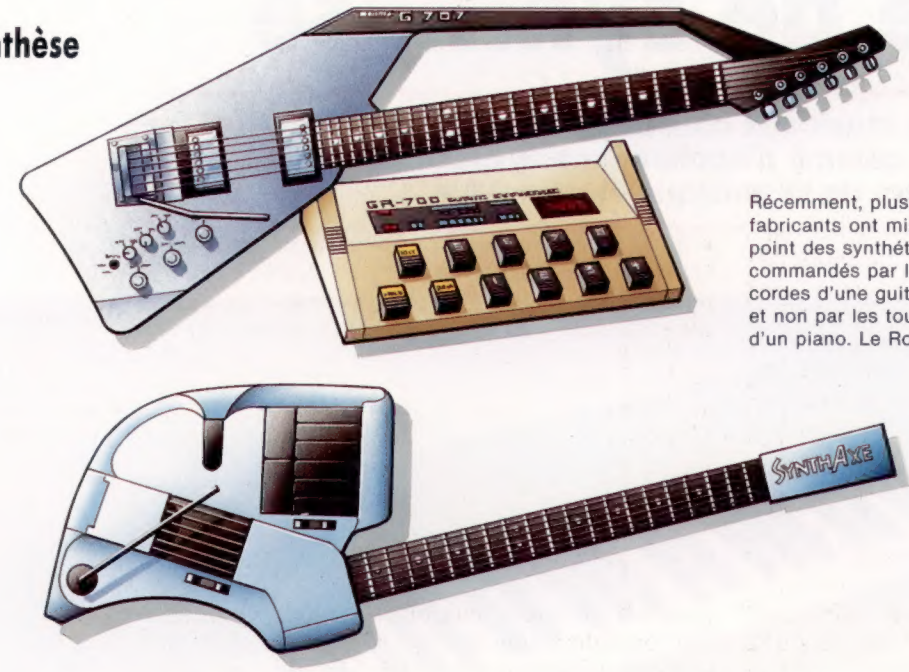
Ce procédé pourrait être étendu par un logiciel jouant un morceau de musique préenregistré sur un synthétiseur interfacé, pendant que défile à l'écran une partition comportant toutes les indications. Ici, la musique et sa notation pourraient être interrompues simultanément et réexécutées à partir d'un numéro de séquence spécifié si l'utilisateur rencontre un problème. De plus, le son musical global pourrait être modifié en changeant les paramètres de commande du synthétiseur — faisant ainsi découvrir à l'utilisateur l'art de l'arrangement.

Après avoir atteint une certaine maîtrise de la lecture musicale, il sera plus facile d'écrire de la musique à l'aide du clavier alphanumérique de l'ordinateur. Il se peut que cela implique l'entrée de données d'exécution, sans réponse sonore immédiate provenant du synthétiseur et sans possibilité de comparer le résultat avec les intentions initiales. Dès que ces connaissances évoluées sont acquises, la notation sur une portée peut être éliminée en faveur d'un autre système comme le MCL *(music composition language)*. En musique électronique, MCL est un mode d'entrée de données plus adéquat, puisqu'il comporte une formulation de caractéristiques propres à la production de sons électroniques. Aucune norme n'a été mise sur pied pour les applications MCL — chaque machine a son propre MCL. La notation sur portée, bien qu'elle soit un guide visuel très pratique, ne peut s'adapter aux nouvelles indications.

Pour de nombreux propriétaires de micro-ordinateurs, la meilleure façon d'exploiter MIDI

Sons et styles nouveaux
Le DX7, l'un des nouveaux synthétiseurs de la gamme DX de Yamaha, offre une méthode de construction des sons, la synthèse FM, réservée jusqu'ici à des machines coûtant des dizaines de milliers de francs. Au lieu de démarrer avec un son existant et de le modifier en le filtrant ou en ajustant des commandes d'enveloppe, le DX7 crée ses propres sons complexes en combinant six formes d'ondes de diverses manières. Le DX7 peut ainsi imiter le son d'instruments acoustiques de façon beaucoup plus précise que les autres synthétiseurs. Le DX7 offre également une commande par souffle qui permet au musicien de souffler dans un dispositif, afin d'ajouter les vibrations du souffle naturel aux sons de trompette et de saxophone.
(Cl. Marcus Wilson-Smith.)

Sons de synthèse



Récemment, plusieurs fabricants ont mis au point des synthétiseurs commandés par les cordes d'une guitare et non par les touches d'un piano. Le Roland GR700 utilise des cordes de guitare standard. Une information sonore complexe est captée par une entrée hexaphonique (six sons) et envoyée au synthétiseur où sont ajoutés les paramètres sonores désirés. Le SynthAxe, un synthétiseur commandé par un 6809, illustre une approche plus complexe de la même technique. Le SynthAxe, encore au stade de prototype, capte les données sonores au moyen d'une connexion électrique établie entre les cordes et les touches. Des capteurs au haut du manche interprètent les variations d'inflexion et de glissement. Un deuxième jeu de cordes et un petit clavier permettent une amélioration supplémentaire du son. (Cl. Kevin Jones.)

au maximum est de comprendre la notation sur portée. Pour la plupart des étudiants en musique, le champ d'étude est toujours basé sur la musique classique européenne. La majorité des musiciens et des professeurs de musique classique associe la musique électronique aux compositeurs d'avant-garde des deux dernières décennies et, sous certains aspects, à la musique pop. On n'admet pas que ces deux domaines puissent être comparés à la musique classique.

Il paraît donc peu probable que le potentiel éducatif de MIDI soit exploré en matière de formation musicale, car certaines connaissances informatiques sont exigées en plus de l'électronique musicale. Il existe pourtant certains cours d'informatique insistant sur les qualités sonores qui pourraient être adaptées pour la formation de joueurs de synthétiseurs MIDI. En outre, la plupart des unités MIDI étant conçues pour établir une interface avec un ou plusieurs synthétiseurs, plusieurs signaux seront nécessaires pour gérer le tout. Même à un niveau élémentaire, l'utilisation de MIDI dans l'enseignement implique le développement d'équipement de studio musical informatisé, et doit impliquer de façon active des étudiants en informatique aussi bien qu'en musique.

Pour les spectacles sur scène, MIDI représente avant tout un moyen d'intégrer plusieurs synthétiseurs, séquenceurs et batteries électroniques dans un système unique facile à commander. Le principal souci des musiciens qui utilisent les séquenceurs de façon intensive dans leurs représentations est la perte de synchronisation. Théoriquement, les groupes utilisant plusieurs synthétiseurs devraient maintenant se produire en toute confiance grâce à MIDI.

Ainsi, de tels groupes n'auront plus à craindre cet effet multisynthétiseurs. Depuis le début des années soixante-dix, un synthétiseur était généralement considéré comme un instrument à clavier muni de nombreux boutons de commande disposés devant les touches. Mais si un synthétiseur à clavier utilise MIDI pour commander un second ou un troisième synthétiseur, le seul clavier indispensable est celui qui commande l'instrument maître.

Comme MIDI devient de plus en plus populaire, d'autres modules de synthétiseur apparaissent sur le marché. Ils ne comportent que les unités de génération sonore et de séquence qui faisaient anciennement partie des instruments à clavier. Ces modules ne présentant que peu d'intérêt visuel, il n'est pas nécessaire de les placer sur scène.

D'autres développements techniques pourraient attirer l'attention en raison de la diminution du nombre de claviers. Il s'agit de la possibilité de commander la synthèse sonore électronique au moyen de guitares et au moyen du souffle appliqué dans des instruments à vent. Comparés à la seule pression sur une touche d'un clavier, le pincement d'une corde ou la vibration d'une anche fournissent une meilleure information acoustique. Si cette information est codée numériquement et transmise par l'intermédiaire de MIDI à un module de synthétiseur situé hors scène, rien n'empêche un joueur de saxophone de commander les instruments électroniques du groupe — même sa batterie électronique. Yamaha a introduit la commande par souffle dans son synthétiseur DX7, et le SynthAxe — une guitare qui vient de faire son entrée sur le marché — ont été conçus afin d'utiliser MIDI pour contrôler la sortie d'un Fairlight.

Cela signifie qu'un son de corde pourrait être produit sur le Yamaha à partir des seules caractéristiques du jeu d'un saxophone, et, similaire-

ment, un son de trombone Fairlight pourrait être articulé en grattant une guitare. Bien qu'aucune de ces nouveautés ne fassent encore partie de la réalité — le SynthAxe est une « guitare » très coûteuse — elles laissent entrevoir l'avenir dans les spectacles.

La présence sur scène de claviers diminuera progressivement. Les instruments à cordes, à vent et à percussion comme les vibraphones seront de plus en plus importants et, l'échantillonnage de sons devenant meilleur marché, les sons acoustiques seront dominants.

Vers la fin des années quatre-vingt, il se peut que les amateurs de musique traditionnelle soient rassurés en voyant — de nouveau — des ensembles de style jazz comprenant guitare, saxophone, basse et batterie. Mais ils peuvent ne pas se rendre compte que le guitariste joue d'un vibraphone invisible et que le joueur de saxophone joue effectivement de la batterie !

Pour de nombreux groupes habitués aux spectacles sur scène, la première expérience dans un studio d'enregistrement perfectionné peut être intimidante. Ils sont confrontés à des instruments musicaux et à des systèmes d'exploitation qu'ils n'ont jamais rencontrés auparavant. Quant au producteur, il peut ne pas connaître particulièrement bien leur travail et leurs intentions. La société d'enregistrement leur demande néanmoins de produire, dans cet environnement peu familier, des versions de leurs « tubes » qui soient meilleures que sur scène. C'est un peu comme engager une troupe semi-professionnelle pour un film à gros budget et s'attendre à un succès commercial immédiat. La transition est parfois difficile, et ce, à tous les égards. Mais, très souvent, les idées originales se perdent dans le labyrinthe des équipements de studio, et le groupe peut connaître un échec qui coûte très cher.

La plupart du temps, le point de rupture survient lorsque le groupe abandonne son propre équipement — en fait sa propre musicalité — et travaille sur les instruments plus appropriés dont il dispose dans le studio. Mais une idée qui marche bien sur un synthétiseur Mini-Moog peut s'effondrer lorsqu'elle est mise en pratique sur un ordinateur d'échantillonnage numérique Fairlight. Si ce type d'échec musical survient assez souvent, il devient de moins en moins justifié d'utiliser un tel studio.

Cependant, si les musiciens du groupe connaissent bien MIDI et si un micro-ordinateur a été utilisé pour stocker les séquences et d'autres données de commande musicale, ils pourront s'adapter facilement aux techniques utilisées dans les studios perfectionnés. Dans un premier stade, il serait possible d'essayer certaines idées, en utilisant une succession d'instruments différents de studio avec le minimum de difficultés, en gardant toujours à l'esprit que les idées et les séquences peuvent être transformées en permutant simplement les synthétiseurs.

Une connaissance de MIDI est également très utile lorsqu'on doit travailler avec des systèmes de studio autres que ceux directement impliqués dans la génération sonore. Une console de mixage à logique électronique, par exemple, dispose d'un ordinateur dédié qui rappellera ou réexécutera toute série de décisions prises dans les dernières étapes d'un enregistrement, c'est-à-dire le mixage. Lorsque toute la musique a été enregistrée sur les vingt-quatre pistes séparées — la guitare sur une piste, les chœurs sur une autre, les voies solos sur trois autres, etc. — commence alors la tâche délicate d'équilibrage et de mixage de tous les éléments. C'est généralement à ce moment que les parties individuelles reçoivent l'« effet » requis pour les faire ressortir du mélange ou les y intégrer. Il peut être nécessaire d'ajuster une certaine réverbération à une seule note de trompette en un seul endroit, mais comme vingt-trois autres événements se produisent en même temps, il est difficile de repérer ce défaut. L'utilisation d'un ordinateur pour traiter de tels incidents lors du mixage est comparable, à une grande échelle, au travail des séquences avec MIDI.

Le code « temps » représente une autre technique, développée initialement pour la synchronisation et pour l'édition vidéo. Le code est comparable à une horloge numérique et à un signal de déclenchement, mais il est enregistré sur bande. Il utilise des mots de 80 bits pour fournir des données de synchronisation lors d'un enregistrement musical accompagnant des séquences vidéo et permet aux événements musicaux d'être synchronisés à la fraction de seconde près avec les séquences vidéo.

Par conséquent, les musiciens ont de bonnes raisons pratiques d'acquérir des instruments compatibles avec MIDI — mais, en plus, MIDI représente une bonne introduction aux systèmes musicaux plus perfectionnés utilisés actuellement. Dans le prochain et dernier article, nous en examinerons un certain nombre.

Science de l'art
Laurie Anderson est une artiste new-yorkaise. En combinant un mélange inusité de sons à des films et à des bandes d'enregistrement son et vidéo, elle a créé un style réellement unique. Dans des chansons comme *O Superman* et *Mr. Heartbreak*, elle se sert d'une gamme très variée d'instruments, allant des clochettes africaines aux instruments électroniques les plus perfectionnés, comme le Vocoder et le Synclavier.

Travail à la pièce

Nous avons vu que l'utilisateur de LOGO peut définir des procédures effectuant des suites de commandes. Ces procédures serviront à en définir d'autres de la même manière que les commandes de base.

Prenons l'exemple d'un puzzle, le Tangram. Il consiste en un carré divisé en sept parties de formes géométriques diverses. Nous utiliserons ces formes pour dessiner un chien (ou du moins essayer...). Nous commençons par définir une procédure par partie géométrique. Ces procédures FORMES sont ensuite intégrées à une autre appelée CHIEN. La tortue devant être positionnée de manière correcte avant chaque tracé de forme, des procédures appropriées, de DÉPLACEMENT1 à DÉPLACEMENT7, seront également définies.

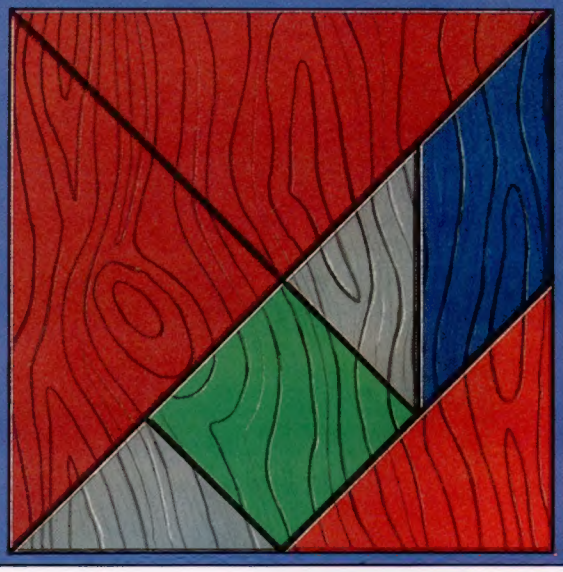
Il serait peut-être plus simple de faire ce dessin par le biais d'une seule longue procédure, en enchaînant les commandes. Notre méthode utilise la structure dite « en pyramide ». Nous avons déjà traité ce sujet. Rappelons simplement qu'il s'agit de décomposer un problème en parties distinctes traitées les unes à la suite des autres. Le principal avantage de cette approche est que le programmeur LOGO peut définir une procédure qui comporte des sous-procédures qui restent à définir. La procédure principale, bien sûr, ne peut être exécutée tant que les sous-procédures n'ont pas été écrites ou du moins remplacées par des procédures virtuelles. Pour voir comment cela se passe, examinons la conception du programme.

Prendre forme

Le puzzle Tangram est constitué de sept formes géométriques que l'on peut combiner en des motifs simples. Voici la liste des procédures LOGO pour les sept formes de base du puzzle, ainsi qu'un programme type qui dessine un chien. La procédure CHIEN commence par tracer un triangle pour les pattes arrière.
(Cl. Kevin Jones.)

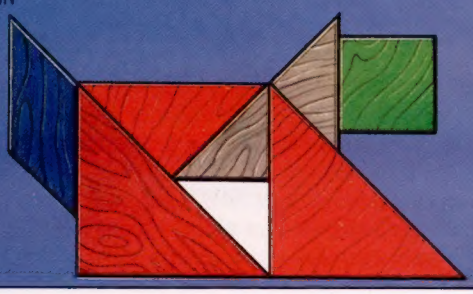
Les procédures Tangram

```
POUR CARRÉ
  RÉPÈTE 4 [AV 25 DR 90]
FIN
POUR PAR
  RÉPÈTE 2 [AV 25 DR 45 AV 35 DR 135]
FIN
POUR TRI1
  AV 25 DR 135 AV 35 DR 135 AV 25 DR 90
FIN
(Il y a deux triangles de cette taille)
POUR TRI2
  AV 35 DR 135 AV 50 DR 135 AV 35 DR 90
FIN
POUR TRI3
  AV 50 DR 135 AV 71 DR 135 AV 50 DR 90
FIN
(Il y a deux triangles de cette taille)
```



Programme « chien »

```
POUR CHIEN
  TRI3 DÉPLACEMENT1 PAR DÉPLACEMENT2
  TRI2 DÉPLACEMENT3 TRI1 DÉPLACEMENT4
  TRI3 DÉPLACEMENT5 TRI1 DÉPLACEMENT6
  CARRÉ DÉPLACEMENT7
FIN
POUR DÉPLACEMENT1
  LP AV 15 LT 45 PP
FIN
POUR DÉPLACEMENT2
  LP DR 45 AV 35 GA 45 AR 35 PP
FIN
POUR DÉPLACEMENT3
  LP GA 45 AR 25 PP
FIN
POUR DÉPLACEMENT4
LP DR 90 AR 25 PP
FIN
POUR DÉPLACEMENT5
  LP AV 50 DR 45 PP
FIN
POUR DÉPLACEMENT6
  LP AV 25 DR 135 AV 5 GA 90 PP
FIN
POUR DÉPLACEMENT7
  CL GA 90 AV 5 DR 45 AR 25 DR 45
  AR 50 GA 90 AR 50 PP
FIN
```



La procédure CHIEN a été écrite en premier, alors même qu'aucune de ses procédures constitutives n'était écrite. Les procédures de dessin des formes géométriques ont ensuite été écrites séparément ; puis les procédures de positionnement. Lorsqu'une nouvelle procédure était écrite, le programme CHIEN était exécuté afin de vérifier si elle s'intégrait parfaitement à l'ensemble. Lorsque LOGO rencontrait une procédure DÉPLACEMENT qui n'avait pas été encore écrite, il s'arrêtait avec un message d'erreur. Le dessin permettait cependant de voir si tout se passait bien ou si la procédure DÉPLACEMENT précédente comportait une erreur.

Ces procédures illustrent un autre point important : chaque procédure FORME (et la procédure CHIEN elle-même) laisse, après exécution, la tortue comme elle l'avait trouvée. Les procédures de ce type sont dites « transparentes à l'état initial » (sans effet sur l'état initial). Cette caractéristique facilite grandement l'assemblage des procédures élémentaires en formes géométriques complexes. Prenez la procédure CHIEN, par exemple : nous savons qu'après avoir tracé un des motifs (une procédure FORME), la tortue reviendra sur la position qu'elle occupait au départ. Nous n'avons donc pas besoin de connaître le fonctionnement des procédures constitutives pour assembler les diverses parties du dessin. A son tour, en rendant la procédure d'ensemble, CHIEN, transparente à l'état initial de la tortue, on se ménage la possibilité de l'intégrer dans une structure plus complexe, un écran rempli de chiens par exemple.

Zone mémoire de logo

Comme vous disposez d'un certain nombre de procédures présentes dans la mémoire de l'ordinateur, voyons d'un peu plus près l'organisation mémoire de LOGO. La zone mémoire de LOGO consiste en une liste de *nœuds* (chacun à 5 octets). Lorsque LOGO est chargé en mémoire, vous avez accès à un nombre de nœuds mémoire compris entre mille et trois mille, selon la machine. Les nœuds sont attribués aux procédures successivement définies.

Les procédures que vous avez définies forment votre zone de travail. Vous pouvez savoir quelles sont les procédures qui y figurent en tapant ATI (Afficher TItres). Pour examiner une procédure en particulier, utilisez AP (Afficher Procédure), par exemple AP CARRÉ. Quand une procédure n'est plus utile, vous pouvez libérer de l'espace en tapant EFFACER ; la commande EFFACER CARRÉ supprime la procédure appelée CARRÉ de la mémoire. Quand vous effacez une procédure, vous libérez les nœuds mémoire utilisés. LOGO les marquera comme tels, mais ne les ajoutera pas à la liste des nœuds disponibles. Il continuera d'utiliser les nœuds disponibles jusqu'au dernier. C'est seulement alors qu'il parcourra la mémoire pour récupérer les nœuds libérés et en constituer une liste de positions mémoire disponibles. Cette démarche est connue sous le nom de « récupération d'espace mémoire » et explique les pauses très brèves de LOGO de temps à autre.

Sauvegarde de procédures

Pour sauvegarder de manière permanente vos procédures sur disque, vous devez stocker la mémoire de travail sous la forme d'un fichier. Si vous prenez comme nom de fichier MESPROCS (MES PROCédureS), vous taperez GARDE «MESPROCS (notez les guillemets avant le nom et non pas après). La mémoire de travail elle-même n'est pas affectée par cette opération. Le fichier est chargé en tapant RAMÈNE «MESPROCS. Les procédures du fichier sont alors définies et ajoutées à la mémoire de travail courante. Lorsqu'une nouvelle procédure reçoit le nom d'une ancienne procédure résidant déjà dans la mémoire de travail, la nouvelle définition efface l'ancienne.

CATALOGUE et EFFACEFICHIER sont d'autres commandes très utiles. CATALOGUE donne la liste des fichiers du disque, et EFFACEFICHIER «MESPROCS, par exemple, effacerait le fichier MESPROCS du disque. Les versions LOGO sur cassettes utilisent des commandes différentes. Il est bon de consulter le manuel d'utilisation pour en savoir plus.

Abréviations

EFFACE	EF
AFFICHE PROCÉDURE	AP
AFFICHE TITRES	ATI

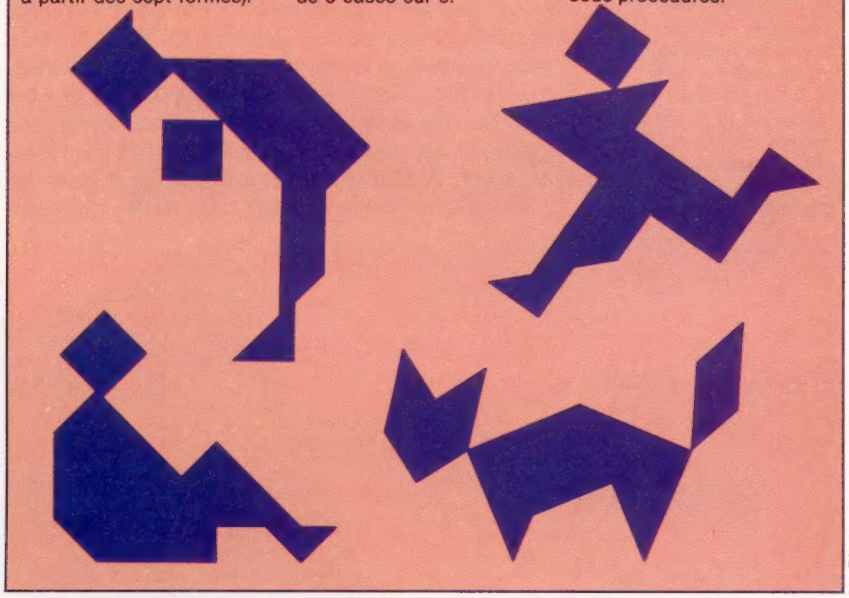
Problèmes de procédures

1) Écrivez des procédures pour les autres dessins représentés ici (vous devrez d'abord résoudre l'énigme de la composition des dessins à partir des sept formes).

2) Écrivez une procédure pour dessiner une maison (un simple triangle équilatéral surmontant un carré).

3) Écrivez une procédure pour dessiner un damier de 5 cases sur 5.

4) Réécrivez la procédure définie précédemment pour tracer une étoile à six branches, de sorte que la nouvelle procédure utilise des sous-procédures.



Liz Dixon

Variantes de logo

Pour toutes les LCSI de LOGO, les noms de procédure doivent être précédés de guillemets s'ils constituent les données en entrée d'une commande AP (Afficher Procédure) ou EFFACE, par exemple AP «CARRÉ, ou EFFACE «CARRÉ.
Pour lire un fichier disque, utiliser LOAD «MESPROCS. Les versions sur bandes pour cassette ou sur microlecteurs utilisent des commandes sensiblement différentes des versions sur disques. Vous devez vous référer au manuel technique approprié.

Bibliothèque de prêts

Comment développer des techniques d'optimisation du temps de programmation? Une méthode est de créer des bibliothèques de programmes que l'on peut inclure dans les programmes principaux.

Les méthodes de programmation structurée que nous avons exposées jusqu'ici peuvent paraître fastidieuses, mais elles permettent vraiment de gagner du temps. Cela est dû au fait que les programmes créés directement au clavier ont tendance à être trop compliqués, avec des algorithmes confus. Ils sont plus longs à écrire et sont davantage sujets à erreurs. En outre, ils prennent davantage de temps à tester et à corriger. Planifier la programmation simplifie l'organisation du programme et des algorithmes.

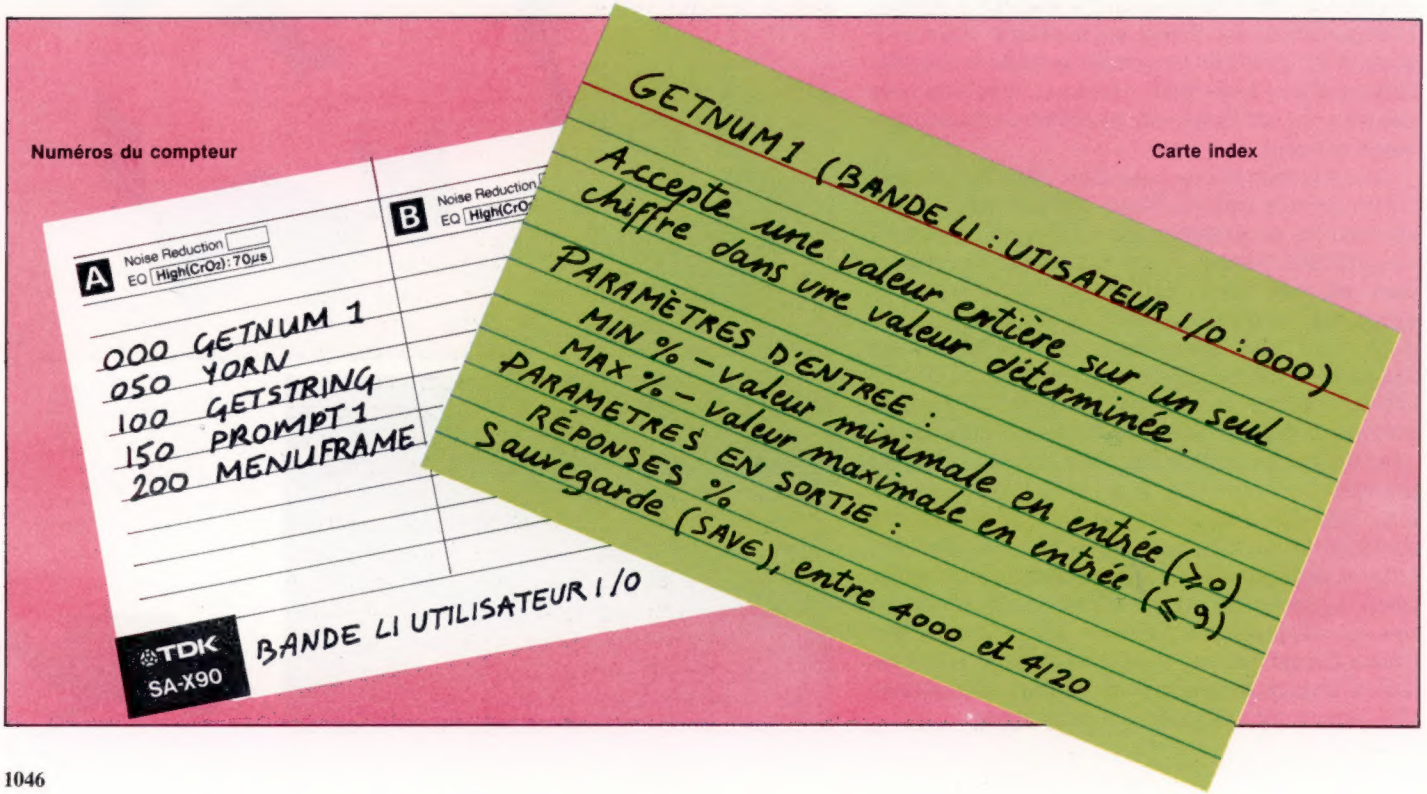
Anticiper sur la programmation permet surtout d'éviter au programmeur d'écrire une structure de commande ou de fichier qui se révèle par la suite non appropriée (par manque de place dans une zone du fichier, par exemple) avec, à la clé, la réécriture de grandes parties du programme.

Il peut être utile pour ceux dont le clavier est de type machine à écrire d'apprendre à taper. En dehors de cela, il y a peu de choses à faire pour accroître la vitesse de saisie d'un programme. Par contre, l'écriture du code du programme peut être, de plusieurs manières, considérablement améliorée. La première est la plus simple : inventez, adoptez et utilisez systématiquement les mêmes conventions pour l'écriture du code. A savoir : un type spécifique pour les noms des variables locales, afin de les différencier des variables du programme principal; commencez chaque sous-programme avec des numéros de lignes se terminant en 000; que chaque sous-programme se termine par RETURN sur la ligne spécifique; que chaque type de sous-programme figure de manière distinctive dans un ensemble donné de lignes (les routines de gestion de fichier entre 9000 et 9999, les utilitaires à partir de 50000, etc.).

L'intérêt d'utiliser ces conventions est multiple. Vous n'avez plus besoin de chercher un peu partout les routines du menu, puisque vous savez qu'elles sont toujours à la même place; vous n'avez pas à vous soucier de savoir si vous avez utilisé le même nom de variable dans le programme principal et dans un sous-programme, puisque le nom d'une variable locale est spécifique et qu'il indique en lui-même qu'il est local.

Ces conventions sont également utiles lors de la création de bibliothèques de programmes. Une bibliothèque de sous-programmes bien organisée peut faire gagner jusqu'à la moitié du temps d'écriture du code. La meilleure façon d'établir une bibliothèque de sous-programmes est de parcourir les programmes existants pour en extraire tous les sous-programmes bien écrits et présentant un certain degré de généralité (routines d'entrée/sortie, de mise à jour de la date, de

Système uniforme
Les bibliothèques de sous-programmes sont sans objet si elles ne sont pas accompagnées d'un système de documentation uniforme. Cela est particulièrement vrai pour les utilisateurs de programmes sur cassettes : avoir à examiner le contenu d'une cassette non documentée en la chargeant et en listant chacun de ses programmes est une tâche vraiment ingrate. (Cl. Mike Clowes.)



conversion majuscules/minuscules, etc.). Chaque routine doit être sauvegardée en tant que fichier séparé. Les fichiers sont groupés selon leur fonction (s'ils doivent être stockés sur cassette, chaque ensemble de fonctions correspondra à une cassette distincte). Le nom des fichiers devra être significatif et en permettre l'identification. Tenez à jour un index des noms de fichiers ou une base de données, avec la description de chaque routine.

Inutile de dire que toutes les routines de bibliothèque auront été testées à fond et corrigées. Elles seront utilisées par des programmes pour lesquels elles n'ont pas été conçues; aussi assurez-vous qu'elles sont capables d'éliminer les saisies de valeurs illégales. Vérifiez également qu'aucune valeur en sortie de routine ne puisse nuire au programme hôte. Vous devrez optimiser les routines et leur incorporer autant de documentation interne que possible, de sorte que vous puissiez ultérieurement comprendre facilement leur fonction. Ajoutez de nouvelles routines à la bibliothèque au fur et à mesure que le besoin s'en fait sentir — il n'y a aucun intérêt à ajouter des routines « au cas où », l'expérience prouvant que c'est le plus souvent du temps perdu. N'oubliez pas de numéroter les lignes des routines selon les conventions adoptées (cela évitera d'avoir à les renuméroter — RENUM — lorsque les routines seront intégrées à un nouveau programme).

Pour pouvoir utiliser la bibliothèque de programmes, il est nécessaire de disposer d'un moyen pour fusionner les routines, de façon à faire un programme complet. Pour celles qui utilisent des langages compilés, un éditeur de liens ou un programme similaire est généralement fourni. L'éditeur de liens prend les modules compilés et les réunit en un programme exécutable. Pour les programmeurs BASIC, le moyen le plus simple de le remplacer (à moins qu'ils ne disposent d'un compilateur) est d'utiliser conjointement les commandes RENUM (renumérotation) et MERGE (fusion). Pour fusionner une routine de bibliothèque avec le nouveau programme, chargez d'abord le programme, décidez de l'implantation de la routine et assurez-vous qu'il y a suffisamment de place sous forme de blocs de numéros de lignes inoccupés. Si nécessaire, renumérotez la routine pour qu'elle tienne dans l'espace qui lui est alloué. Utilisez ensuite la commande MERGE pour fusionner les deux programmes, vérifiez que tout est en place et fonctionne convenablement, et sauvegardez (SAVE) le nouveau programme qui comporte maintenant la routine de bibliothèque.

Efforts de groupe

Les utilisateurs d'ordinateurs personnels travaillent souvent en groupe pour écrire les programmes. Tout ce que l'on a dit à propos de la conception des programmes et de l'efficacité du programmeur est particulièrement vrai dans ce cas. En réalité, la plupart de ces idées, ainsi que le concept de programmation structurée, ont été developpés pour répartir la charge de travail en la divisant, dans le cadre de projets commerciaux de programmation. Ainsi, plusieurs programmeurs pouvaient travailler simultanément au même programme sur des parties bien distinctes, le tout faisant un programme cohérent et opérationnel. Pour les programmeurs BASIC qui travaillent de la sorte, l'essentiel est de s'entendre sur les conventions utilisées pour le code. En supposant que l'organisation générale du programme ait été déterminée, le programmeur d'un module déterminé devra connaître :

1) la composition de fichiers et leur organisation;
2) les conventions retenues pour le nom des variables. Les variables les plus importantes, celles qui sont utilisées d'un bout à l'autre du programme, doivent être nommées d'avance. Une convention doit être adoptée pour les variables locales. Les variables qui sont communiquées d'un module à l'autre seront soit déterminées d'avance, soit rendues nécessairement uniques, en ajoutant le numéro du module d'origine comme suffixe, par exemple;
3) les routines accessibles au groupe de programmeurs, leur format, le nom de leurs variables, leur fonction et leur degré de fiabilité;
4) l'organisation des routines de gestion d'erreurs (pour savoir par exemple si chaque routine gère ses propres erreurs ou met un drapeau d'erreur traité ensuite par la routine de commande);
5) la fonction exacte de tous les modules écrits;
6) la fourchette exacte des données admises pour chaque module, ainsi que le type des données, en entrée et en sortie.

Tout cela suppose une phase de préparation longue et minutieuse, avec de nombreuses rencontres entre les programmeurs pour se mettre d'accord sur une stratégie. Il en résulte évidemment que la phase de programmation sera très courte en comparaison.

La phase des tests, y compris le test des programmes produits par le groupe, viendra après.

Fusion

- Le Spectrum comporte la version la plus rapide de la commande : elle fusionne le fichier nommé avec le programme en mémoire; en cas de conflit de lignes, celles qui entrent effacent les anciennes.
- La commande *SPOOL du BBC Micro permet de créer des versions ASCII des fichiers du programme, d'écrire ensuite un programme BASIC (ou d'utiliser un traitement de texte) pour accéder à ces fichiers, une ligne à la fois. La fusion des deux fichiers se fait sur un troisième fichier ASCII qui sera transcrit en un programme par la commande *EXEC.
- Avec le Commodore :

OPEN 1, 1 :CMD1:LIST:PRINT#1:CLOSE1 crée sur bande un fichier ASCII sans nom du programme en mémoire. LOAD charge l'autre programme, et lui ajoute une routine de saisie (INPUT) et d'affichage à l'écran des lignes du fichier ASCII. Le programme est stoppé et envoyé à l'écran (RETURN), fusionnant les deux programmes.

Offre sérieuse

Apple Computer se devait de relancer son fleuron, l'Apple II, qui réunit toute une gamme de micros basés sur le microprocesseur 6502. C'est ainsi que la famille des Apple II*c* est née.

Le succès du Macintosh et l'accroissement de la concurrence ont jeté une ombre sur Apple II et sa tribu... De nombreux spécialistes prédisaient l'extinction de l'Apple II, malgré l'insistance de ses créateurs à le maintenir. L'Apple II*c* a donc été lancé pour affirmer cette confiance, et il est accompagné d'extensions logiciel et matériel. Les nouveaux produits seraient destinés à prolonger la vie de l'Apple II de manière importante, au moins pour trois ans !

Il est vrai que sous ses diverses formes (II, II+, II*e*), l'Apple II a beaucoup contribué à développer le marché de l'ordinateur personnel. Il a permis à Apple d'atteindre un chiffre de ventes total supérieur à un milliard de dollars ! Il y a environ deux millions d'ordinateurs Apple dans le monde, en majorité aux États-Unis. Son prix est un peu trop élevé en Europe pour un ordinateur personnel. Les très nombreux utilisateurs de ce micro de par le monde sont très attachés à leur machine et très satisfaits de ses performances. La fidélité de ses utilisateurs est donc le meilleur atout d'Apple.

Extension portable
Le nouvel Apple, l'Apple II*c*, est une extension portable de l'Apple II. Il offre 128 K de RAM, un affichage quatre-vingts colonnes, de nombreuses interfaces et un lecteur incorporé. Il est montré ici avec le moniteur vert phosphorescent qui est livré en option.
(Cl. Chris Stevens.)

La dernière incarnation de l'Apple II est le II*c* (« *c* » pour « compact »). Il est en effet plus petit, environ de moitié, bien qu'il contienne un lecteur 5 1/4 pouces demi-hauteur sur le côté. L'Apple II*c* pèse 3,400 kg seulement et entre donc dans la catégorie des micros portables. L'utilisation prévue par ses concepteurs consiste à en faire un ordinateur de gestion pendant la journée au bureau, et à le rapporter chez soi le soir comme ordinateur personnel. Il dispose d'une poignée moulée dans son boîtier de plastique. Il est accompagné de plusieurs connexions, pour être utilisé avec un moniteur polyvalent ou RVB (rouge-vert-bleu) au bureau, ou avec un téléviseur à la maison.

Contrairement aux modèles précédents, l'Apple II*c* est un système clos, dépourvu de connecteurs d'extensions. Les options les plus importantes ont été incorporées à la machine. Ces dernières comprennent les ports d'affichage TV et moniteur, un port manette de jeu également destiné à la « souris » en option, un port modem, un port imprimante, une prise sortie audio et un connecteur pour un second lecteur de disque. Les interfaces sont identifiées par de petites figures symboliques. L'Apple II*c* comporte également l'affichage standard sur quatre-vingts colonnes et dispose de 128 K de RAM. La plupart de ces caractéristiques étaient optionnelles sur l'Apple II*e* et supposaient sur cette machine l'adjonction d'au moins trois cartes d'extensions.

L'Apple II*c* dispose d'un clavier de soixante-trois touches AZERTY, de même disposition que celui de l'Apple II*e*. La touche « reset » (réinitialisation), a été déplacée jusqu'à l'extrême gauche du clavier, et deux petits commutateurs ont été ajoutés à côté. Celui de gauche bascule l'affichage d'un mode à l'autre (80 colonnes/40 colonnes). Le manuel utilisateur recommande le mode 40 colonnes avec le téléviseur, et le mode 80 colonnes avec le moniteur. Cependant, certains logiciels existants afficheront seulement en mode 40 colonnes. Le deuxième interrupteur bascule le mode caractères américains sur le mode caractères européens qui correspond au clavier, et inversement. Cela peut être utile lorsqu'on a besoin d'un caractère qui n'est pas sur le clavier (tel que « # » qui est remplacé sur le clavier européen par le signe £). Deux diodes lumineuses à droite du clavier indiquent que l'appareil est sous tension et que le lecteur est en cours d'utilisation.

Lorsque vous allumez l'Apple II*c*, le lecteur de disque se met automatiquement en mouvement dans l'attente d'un disque. Il restera ainsi jusqu'à

introduction d'un disque ou jusqu'à la frappe simultanée des touches « reset » et « contrôle ». En l'absence de disque, le BASIC Applesoft est chargé depuis la ROM. Cette version du BASIC n'a pratiquement pas été changée depuis l'arrivée de l'Apple II+. Elle est pratiquement semblable aux premières versions du BASIC Microsoft standard. Son emploi permet d'exploiter des logiciels Applesoft écrits pour des versions antérieures de la machine. En contrepartie, de nombreuses constructions syntaxiques de programmation, présentes sur des versions plus avancées du langage BASIC, manquent au BASIC Applesoft. Ce dernier n'a pas la fonction RANDOMIZE, le dispositif AUTO numérotation des lignes, IF...THEN...ELSE ou encore la commande WHILE. Il n'a pas non plus les commandes graphiques CIRCLE et PAINT destinées à la programmation graphique.

Lorsqu'un disque est présent dans le lecteur, l'Apple II*c* utilise soit le système d'exploitation DOS 3.3 (disques pour Apple II+ et Apple II*e*), soit le nouveau système d'exploitation PRODOS. Il s'agit d'un dérivé du système d'exploitation qu'Apple avait conçu pour son premier micro de gestion, l'Apple III. PRODOS dispose d'une structure hiérarchisée (arborescente) des fichiers. Les fichiers disque sont stockés un peu de la même manière que des dossiers dans une armoire. Ainsi, tous les fichiers ayant trait par exemple au projet ZED sont classés sur le disque sous la dénomination ZED. Les fichiers comptables du projet ZED, les coûts, les ventes, les profits peuvent être réunis en un groupe appelé « comptes ».

Avec un système de classement manuel, si vous vouliez trouver le fichier VENTES, il vous fallait prendre d'abord le dossier ZED, l'ouvrir, ouvrir ensuite le dossier COMPTES, et passer en revue plusieurs sous-dossiers pour en arriver au fichier étiqueté VENTES. Avec PRODOS, vous y parvenez via un « nom-aiguillage » qui indique les embranchements à suivre et liste dans l'ordre les noms de fichiers à ouvrir. Pour l'exemple que nous avons vu plus haut, il suffit donc de taper :

```
/ZED/COMPTES/VENTES/
```

Les « noms-aiguillages » peuvent comporter jusqu'à soixante-quatre caractères. Cette démarche peut sembler compliquée, mais il suffit d'un peu d'entraînement pour s'y habituer. A long terme, elle représente une simplification importante pour l'organisation et la gestion des fichiers disque. Des systèmes d'organisation arborescente de fichiers tels que PRODOS sont utilisés aussi sous MS-DOS, le système d'exploitation de l'IBM PC.

L'affichage de l'Apple II*c* représente également une amélioration par rapport aux modèles précédents. Outre deux options Texte (24 lignes par 80 colonnes ou par 40 colonnes), il comprend trois écrans graphiques : 40 × 40 (faible résolution), 280 × 192 (haute résolution) et 560 × 192 (appelé double-haute résolution). Seize couleurs sont disponibles. Un écran moniteur à phosphore vert est proposé en option à un prix intéressant. Un écran plat à cristaux liquides sera

Prudence, prudence...

Chris Steyaert

Apple III

Lisa

Réalisant l'importance du marché de la micro-informatique professionnelle, Apple a développé l'Apple III, une machine très performante de gestion qui utilise deux microprocesseurs 6502 pour 512 K de mémoire. Le système d'exploitation s'appelle SOS (Sophisticated Operating System), et lui permet de communiquer avec une sauvegarde-disque dure et d'organiser les fichiers de manière hiérarchisée. Cette arborescence a été à la base du système d'exploitation de Lisa. De nombreuses caractéristiques du SOS ont été reprises dans le MS-DOS du IBM PC. Malheureusement, l'Apple III a connu des difficultés techniques peu après son lancement, ce qui a nui à sa réputation. Apple a repris et remplacé toutes les machines défectueuses, mais n'a pu corriger cette mauvaise réputation. L'Apple III a été retiré de la vente.
Lisa a fait son apparition au printemps 1983 et fut le premier micro personnel de grande diffusion à utiliser des logiciels intégrés et à fenêtre. Son système d'exploitation est basé sur des images et non sur des mots, et se commande au moyen d'un périphérique manœuvré à la main, la « souris ». L'intérêt pour Lisa fut tel que les actions Apple passèrent en quelques mois de 24 $ à 60 $. Mais Apple avait mal ciblé son marché. Il était destiné aux cadres, et si certains s'y intéressèrent, la plus grande partie des ventes provinrent de petites sociétés spécialisées en publicité, conception graphique et relations publiques. Le Lisa ne se vendit pas aussi bien que prévu, et l'action Apple chuta à 17 $ à nouveau en quelques mois. Lisa a été récemment remplacé par Lisa II, beaucoup moins cher. Sous le contrôle de John Scully, qui vient de chez Pepsi-Cola, la société Apple, qui a toujours su innover technologiquement, entreprend de tirer les enseignements de ses échecs commerciaux. Le Macintosh se vend maintenant en quantités énormes (350 000 unités par an) à travers le monde, recouvrant une partie des frais de développement de Lisa/Mac, et l'Apple II*c* semble apporter une vie nouvelle au gagne-pain d'Apple, l'Apple II.

prochainement commercialisé. Ce dernier sera construit pour Apple par Sharp et affichera 20 lignes de 80 colonnes. Il sera proposé à un prix cinq fois supérieur au prix de l'écran vert. Lorsque l'Apple IIc disposera (c'est pour bientôt), d'accus rechargeables, il sera vraiment un portable complet.

Le principal avantage de l'Apple IIc tient à sa vaste gamme de logiciels de base. En effet, dix-sept mille programmes ont été écrits pour l'Apple II. Bien que de nombreux programmes soient restés confinés aux États-Unis, vous pouvez être sûrs que tout ce que l'on peut faire avec un Apple II a déjà été écrit.

Parmi ceux-ci, des logiciels de jeux très célèbres (CHESS 7.0, ZORK, simulateur de vol Microsoft, jeu de construction Pinball); de nombreux traitements de texte, tableurs et bases de données; des programmes de comptabilité et de création graphique; des programmes scientifiques et des didacticiels allant depuis l'apprentissage de la lecture jusqu'aux calculs mathématiques très élaborés.

En plus des logiciels existant sur Apple II et Apple IIe, Apple propose un programme appelé Appleworks, intégrant un traitement de texte, un tableur et une base de données avec fenêtres. Appleworks est un logiciel très sophistiqué et d'un abord facile. L'Apple IIc est livré avec une disquette d'introduction à Appleworks, mais qui ne contient pas le programme. Dans l'esprit d'Apple, l'utilisateur alléché finira bien par l'acheter (dans les 1 500 F). D'autres disques fournis avec le système d'exploitation servent ainsi d'introduction à divers programmes, comme Apple-Logo. Il y a aussi une introduction interactive au système de base, et une introduction très simple à la programmation en BASIC. Le disque des utilitaires (système PRODOS) est fourni. MousePaint, le programme pour piloter la « souris » qui dessine, est basé sur MacPaint, programme graphique plus sophistiqué.

Ensemble de logiciels Apple
L'Apple IIc est livré avec cinq disquettes qui décrivent le fonctionnement de la machine, la programmation BASIC, et annoncent des programmes d'applications. La dernière comporte les utilitaires et le système d'exploitation PRODOS qui organise les fichiers.

Un guide de l'utilisateur de 142 pages expose de manière brève et claire l'exploitation du système et l'usage des 5 disques livrés avec la machine. Les manuels sont bien écrits et bien illustrés. Ils s'adressent de toute évidence à l'utilisateur novice.

L'Apple IIc est d'un aspect très attrayant. Apple a laissé tomber le plastique beige utilisé pour l'Apple II, le Macintosh et Lisa, pour une finition d'un blanc brillant. Les bandes du genre « voiture de course » sur le dessus du boîtier servent à la ventilation des circuits.

L'Apple IIc, comme ses prédécesseurs, est un grand micro professionnel. Avec l'écran plat à cristaux liquides et les accus rechargeables, qui sont attendus pour bientôt, il devrait devenir un portable de grande réputation. Il ne manque qu'un prix plus modique pour en faire un ordinateur personnel et familial.

MousePaint
La « souris » est disponible sur le II+, le IIe et le IIc, pour 1 000 F environ avec MousePaint. Ce programme est basé sur MacPaint, même s'il s'agit d'une version dérivée. Il vous permet de dessiner facilement avec la souris et utilise pleinement le mode haute résolution. Pour utiliser la souris avec d'autres versions de l'Apple II, il vous faut une interface supplémentaire qui se loge dans un des connecteurs d'extension.

Sortie vidéo polyvalente

Port imprimante RS232

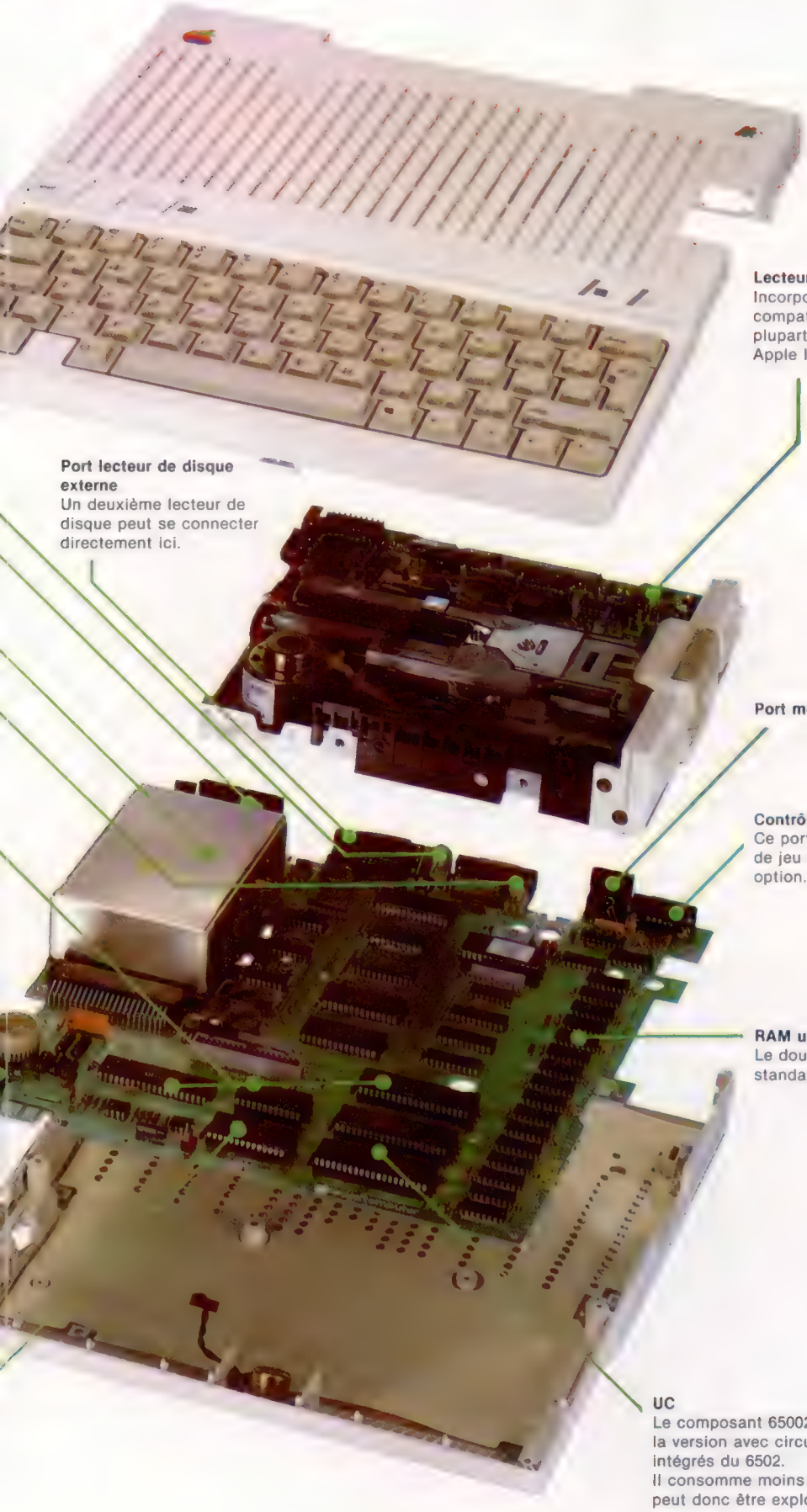
Alimentation
Interne, de 12 V, nécessite pourtant un transformateur pour le 220 V.

Sortie vidéo RVB
Avec un adaptateur, la connexion est possible avec un téléviseur standard.

Contrôleurs d'entrée/sortie
Composants contrôlant le clavier et les ports d'entrée/sortie.

Prise audio
Connexion possible avec un amplificateur Hi-Fi externe.

ROM
Contient le BASIC Applesoft et les routines de service.



Lecteur de disque
Incorporé, de 143 K et compatible avec la plupart des disquettes Apple II+ et IIe.

Port lecteur de disque externe
Un deuxième lecteur de disque peut se connecter directement ici.

Port modem RS232

Contrôle manuel
Ce port gère une manette de jeu ou la souris en option.

RAM utilisateur 128 K
Le double de la mémoire standard de l'Apple IIe.

UC
Le composant 65002 est la version avec circuits intégrés du 6502.
Il consomme moins et peut donc être exploité avec des accus.

APPLE IIc

PRIX
★ ★ ★ ★

DIMENSIONS
305 × 292 × 64 mm.

UC
6502 à 1 MHz.

MÉMOIRE
128 K RAM, 16 K ROM.

ÉCRAN
24 ou 40 lignes de 80 caractères. Trois modes d'affichage graphique avec une résolution maximale de 560 × 192 pixels en 16 couleurs.

INTERFACES
Port à 9 broches pour manette de jeu, également pour la « souris », port modem RS232, sortie RVB ou TV, sortie vidéo polyvalente, port lecteur de disque externe et port imprimante RS232.

LANGAGES DISPONIBLES
BASIC Applesoft résidant en ROM, LOGO, PASCAL et FORTRAN.

CLAVIER
63 touches, type machine à écrire avec bloc-curseur de 4 touches et un jeu de caractères internes.

DOCUMENTATION
Guide de l'utilisateur très bien illustré de photos couleurs, et très compréhensible, livré avec la machine. Conçu pour le débutant. Deux manuels pour la mise en route, tournant avec les utilitaires.

FORCES
Toute sa force est dans sa compatibilité avec l'Apple II. Il peut donc exploiter les quelque 17 000 logiciels développés sur les autres Apple.

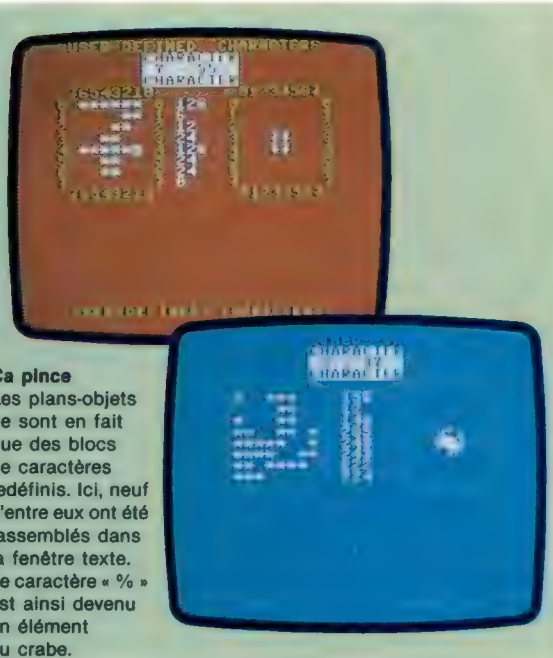
FAIBLESSES
Le BASIC Applesoft, qui n'a pas changé de manière significative en six ans, manque de souplesse. Le prix du IIc le met hors de portée des utilisateurs d'ordinateurs individuels.

Gutenberg 64

Avec de la patience, on surmonte la faiblesse du BASIC du Commodore 64. Le programme que nous vous présentons ici vous permettra de créer votre propre jeu de caractères.

Les possibilités graphiques et sonores du Commodore 64 sont énormes; bien des jeux vendus dans le commerce en sont la preuve. Pourtant, il ne dispose d'aucune commande spécifique qui gérerait le son et la couleur, comme les BEEP, DRAW, INK, PAPER, etc. Il s'ensuit que la plupart des programmes BASIC rédigés à son intention restent extrêmement limités de ce point de vue; les meilleurs d'entre eux sont remplis d'instructions POKE et DATA — le nôtre ne fait pas exception à la règle. Il a pourtant l'avantage de vous faciliter la tâche : si vous désirez redéfinir vos caractères, il vous suffira de les dessiner sur l'écran, plutôt que de POKEr des valeur en RAM. C'est le programme qui s'en chargera.

Nous avons déjà abordé le problème de la création de nouveaux caractères sur le Commodore 64. Les opérations préliminaires, essentielles, sont effectuées par la sous-routine qui commence à la ligne 61000. La mémoire dont l'utilisateur peut faire usage se voit déplacée vers le bas. Sa limite supérieure, ordinairement fixée à l'adresse 40959, passe en 14335. Le jeu de caractères majuscules résidant en ROM (à partir de l'adresse 53248), et qui occupe 2 K, est alors recopié en RAM (à partir de l'adresse 14336), où il peut être manipulé par le biais d'instructions PEEK et POKE. Enfin, la puce VIC (Video Interface Chip) est orientée vers le nouvel emplacement.



Sept d'un coup
On aperçoit ici, dans la fenêtre d'édition située à gauche, une version redéfinie du chiffre 7. On en a élargi le pied, lui ajoutant au passage une barre transversale (usage propre à l'Europe, mais inconnu des Anglo-Saxons). Quatre copies de ce caractère ainsi revu apparaissent dans la fenêtre « texte » sur la droite. Les changements y sont bien visibles; seul le chiffre placé dans le cadre blanc en haut de l'écran a gardé sa forme originale.

Ça pince
Les plans-objets ne sont en fait que des blocs de caractères redéfinis. Ici, neuf d'entre eux ont été rassemblés dans la fenêtre texte. Le caractère « % » est ainsi devenu un élément du crabe.

Ian McKinnell

Une fois ces préliminaires accomplis, la routine d'initialisation affiche deux « fenêtres » à l'écran, puis le contrôle passe à la routine d'entrée de la ligne 2500. Elle gère toute frappe du clavier et met en place un curseur clignotant dans la fenêtre d'« édition » (à gauche). C'est là qu'est affiché (suffisamment agrandi) le caractère particulier sur lequel vous travaillez. Les valeurs des 8 octets qui le définissent sont indiquées juste à côté.

Les quatre touches de fonction (F1, F3, F5, F7) permettent de déplacer le curseur avec précision à l'intérieur de cette fenêtre. Toute case au-dessus de laquelle il se trouve correspond à un bit de l'un des 8 octets. Elle peut être « allumée » ou « éteinte » grâce à F2 (qu'on obtient par SHIFT+F1). Chaque fois que cela se produit, le caractère, où qu'il se trouve sur l'écran, est instantanément modifié.

En appuyant sur F4 (SHIFT+F3), vous pourrez passer à un autre caractère. Chacun d'eux est défini par son code d'écran (la valeur à POKEr) tel qu'il est donné dans l'appendice F du manuel de l'utilisateur. Il ne faut pas le confondre avec le code CHR$ correspondant, bien qu'il y ait des rapports entre les deux. Il a été choisi ici parce qu'il est d'emploi plus facile; en effet, c'est sous cette forme que les caractères sont stockés en mémoire.

F6 (SHIFT+F5) provoque l'apparition du caractère en cours d'édition dans la fenêtre « texte », située à droite. Il est de taille réelle, et sa position est fonction de celle du curseur de la fenêtre d'édition; si ce dernier est en haut et à gauche, la copie du caractère aura le même emplacement dans la fenêtre texte.

Enfin, appuyer sur le point d'exclamation provoque l'arrêt du programme, que vous pouvez cependant relancer en tapant CONT. Une fois que vous en avez terminé, il est possible de taper NEW, puis de charger un autre programme, sans que le nouveau jeu de caractères en soit perturbé. Pourtant, deux problèmes demeurent : d'une part, abaisser la limite supérieure de la mémoire ne laisse plus que 12 K à la disposition d'un nouveau programme; d'autre part, tout votre travail sera réduit à néant dès que vous mettrez l'ordinateur hors tension. Nous examinerons ces difficultés dans un article ultérieur. En attendant, sans doute pourriez-vous noter les définitions des caractères que vous avez mis au point et vous en servir, si vous le désirez, dans le précédent programme de création de caractères sur Commodore 64.

Définir les caractères

```
19 REM**********C64***************
20 REM * GENERATEUR DE CARACTERES *
21 REM**********C64***************
50 POKE 52, 56 : POKE 56, 56 : CLR
60 PRINT CHR$(147) "UN PETIT INSTANT S.V.P."
70 GOSUB 61000 : REM COPIE JEU CARACTERES
100 GOSUB 1000 : REM INITIALISATION
120 FOR CT=0 TO 1 STEP 0
140 GOSUB 2500 : REM ENTREES
160 GOSUB 3000 : REM VERIFICATION
180 ON PT GOSUB 3500, 4000, 4500, 7000
200 NEXT CT
900 END
997 REM**********************
998 REM*  INITIALISATION   *
999 REM**********************
1000 DIM BD(8,8), C$(2,2), OF(2,7), CX(4)
1040 SH$=CHR$(19) : SC$=CHR$(147) : R$=CHR$
(18) : N$=CHR$(146) : CD$=CHR$(17)
1060 P$=CD$+CD$+CD$+CD$ : P$=P$+P$+P$+P$+P$ : P$=
P$+P$ : P$=SH$+P$
1080 C1$=CHR$(144) : C2$=CHR$(5)
1200 REM -----INITIALISATION ECRAN -----
1210 SO=1024 : PRINT SC$, C1$
1220 RO=4 : CO=3 : RL=8 : CL=8 : OF=16
1230 Z$= "CARACTERES REDEFINIS"
1240 C=4 : GOSUB 2100 : R=24 : GOSUB 2100
1250 L$=" 76543210 " : S$="        "
1260 Z$=L$ : R=RO : C=CO : GOSUB 2100
1270 C=CO-1 : FOR R=RO+1 TO RO+8
1280 Z$=STR$ (R-RO-1) : Z$=Z$+S$+Z$
1290 GOSUB 2100 : C=C+OF : GOSUB 2100
1300 C=C-OF : NEXT R
1310 C=CO : Z$=L$ : GOSUB 2100
1320 L$=" 01234567 "
1330 Z$=L$ : R=RO : C=CO+OF : GOSUB 2100
1350 C=CO+OF : R=RO+CL+1 : GOSUB 2100
1370 PRINT C2$
1400 C$(1,1)= R$+ " "+N$ : C$ (1,2)= " "
1410 C$(2,1)=R$+"+"+N$ : C$(2,2)="+"
1420 REM -----SORTIES CURSEUR -----
1440 DATA 0,-1,+1,0,-1,0,0,+1
1450 FOR K=1 TO 2 : FOR L=1 TO 4:
1480 READ OF(K,L) : NEXT L, K
1500 REM -----CONVERSION CARACTERES -----
1520 DATA 64,0,32,64
1540 FOR K=0 TO 3 : READ CX(K) : NEXT K
1620 RP=1 : CP=1 : CN=1 : GOSUB 6000
1990 RETURN
1997 REM **********************
1998 REM * PLACE CURSEUR@ R, C *
1999 REM **********************
2000 PRINT LEFT$(P$, R+1)TAB(C);
2020 RETURN
2097 REM **********************
2098 REM *  IMPRIME Z$@ R, C   *
2099 REM **********************
2100 PRINT LEFT$(P$, R+1) TAB(C)Z$ ;
2120 RETURN
2497 REM **********************
2498 REM * FLASH CURSEUR@ RP, CP *
2499 REM **********************
2500 CF=1+BD (RP,CP) : R=RP+RO : C=CP+CO
2540 FOR LP=0 TO 1 STEP 0
2560 FOR CS=1 TO 2: DE=10 : GOSUB 2800
2580 GET GT$
2600 IF GT$<>" " THEN LP=1 : CS=2
2620 Z$=C$(CF,CS) : GOSUB 2100
2640 DE=10 : GOSUB 2800
2660 NEXT CS,LP : RETURN
2797 REM **********************
2798 REM * BOUCLE DE RETARD  *
2799 REM **********************
2800 FOR NN=1 TO DE : NEXT : RETURN
2997 REM **********************
2998 REM * VERIFICATION ENTREE *
2999 REM **********************
3000 IF GT$="!" THEN R=18 : C=0 : GOSUB 2000 : STOP
3040 GT=ASC(GT$)-132 : F=2*INT(GT/2)
3060 IF (GT<1) OR (GT>8) THEN PT=0 : RETURN
3080 IF GT<5 THEN PT=1 : RETURN
3100 PT=GT-3
3490 RETURN
3497 REM **********************
3498 REM * MOUVEMENT CURSEUR *
3499 REM **********************
3500 NY=RP+OF(2,GT) : NX=CP+OF(1,GT)
3540 IF (NY<1) OR (NY>RL) THEN RETURN
3560 IF (NX<1) OR (NX>CL) THEN RETURN
3580 RP=NY : CP=NX
3620 RETURN
3997 REM **********************
3998 REM * ALLUME/ETEINT CASE *
3999 REM **********************
4000 TG=1-BD(RP,CP) : Z$=C$(1+TG,2)
4040 R=RO+RP : C=CO+CP : GOSUB 2100
4060 BD(RP, CP)=TG : MP=NCGEN+CN+8-1
4120 PE=PEEK(MP+RP)
4140 PE=PE-(TG*2-1)*(2 (8-CP))
4160 POKE(MP+RP), PE : GOSUB 6500 : RETURN
4497 REM **********************************
4498 REM * DEFINITION NOUVEAU CARACTERE *
4999 REM **********************************
4500 FOR K=1 TO 1
4540 Z$=R$+"  CHANGE  CAR   "
4550 R=14 : C=7 : GOSUB 2100
4560 Z$="NOUVEAU NUMERO "
4570 R=15 : GOSUB 2100
4580 C=19 : GOSUB 2000 : INPUT A$
4600 CN=VAL(A$) : PRINT C2$
4620 IF (CN<0) OR (CN)127) THEN K=0
4640 NEXT K : GOSUB 6000
4660 Z$="                " : C=7
4670 R=14 : GOSUB 2100 : R=15 : GOSUB 2100
4680 RETURN
5997 REM **********************
5998 REM * AFFICHAGE CARACTERE *
5999 REM **********************
6000 MP=NCGEN+CN+8-1
6040 FOR RP=1 TO 8 : PE=PEEK(MP+RP) : Z$=""
6060 FOR CP=8 TO 1 STEP -1 : N=INT(PE/2)
6080 Q=PE-2*N : BD(RP,CP)=Q : PE=N
6100 Z$=C$(Q+1,2)+Z$ : NEXT CP
6120 R=RO+RP : C=CO+1 : GOSUB 2100
6130 NEXT RP
6140 X$=CHR$(CN+CX(INT(CN/32))
6150 Z$=R$+"CARACTERE" : R=1 : C=11
6160 GOSUB 2100 : R=R+2 : GOSUB 2100
6170 Z$=" "+X$+" =     "
6180 R=R-1 : GOSUB 2100
6190 C=C+4 : Z$=STR$(CN)+N$ : GOSUB 2100
6220 RP=1 : CP=1
6490 GOSUB 6500 : RETURN
6497 REM **********************************
6498 REM * AFFICHAGE VALEURS OCTETS *
6499 REM **********************************
6500 C=CO+CL+2 : FOR R=RO+1 TO RO+8
6560 Z$=STR$(PEEK(MP+R-RO))+"  "
6580 GOSUB 2100 : NEXT : RETURN
6997 REM **********************
6998 REM * PLACEMENT CARACTERE *
6999 REM **********************
7000 Z$=X$ : C=C+OF : GOSUB 2100 : RETURN
60997 REM **********************************
60998 REM * DEPLACEMENT JEU CARACTERES *
60999 REM **********************************
61000 CGEN=53248 : NCGEN= 14336
61020 ITRPT=56334 : IOPT=1 : PO=53272
61125 RETURN
61050 POKE IT,PEEK(IT)AND254
61100 POKE IO,PEEK(IO)AND251
61150 FOR J=0 TO 2047
61200 POKE(NCGEN+J), PEEK(CGEN+J)
61250 NEXT J
61300 POKE IO,PEEK(1@)OR4
61350 POKE IT,PEEK(IT)OR1
61400 POKE OP,(PEEK(PO)AND240)OR 14
61500 RETURN
```

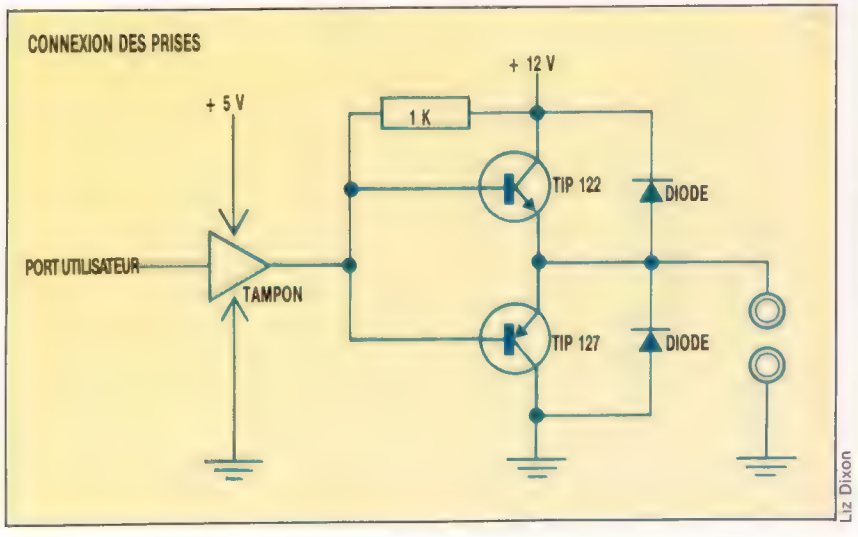
Interrupteur

Nous poursuivons notre cours en expliquant comment construire un dispositif d'interface permettant à votre ordinateur de commander de petits moteurs électriques et des ampoules.

Le boîtier
Voici le produit final montrant la prise spéciale à 12 voies et les conducteurs d'entrée/sortie. Le boîtier doit être coupé avec soin afin que la carte ne puisse bouger dans le boîtier lors de l'insertion d'une fiche dans un connecteur.
(Cl. Ian McKinnell.)

Cette interface permettra à votre ordinateur de commander des dispositifs à basse tension consommant peu d'électricité. La sortie des 4 bits du port utilisateur est stockée temporairement par la même puce tampon que nous avons utilisée précédemment dans la construction du boîtier tampon. La sortie provenant de ce dispositif sert à mettre sous tension les transistors qui sont en mesure de commander des tensions d'alimentation plus élevées que celles que le tampon peut gérer. La configuration complémentaire de transistors permet à la sortie d'alimenter une charge ou d'en supprimer le courant. Les deux diodes placées à la sortie protègent les transistors contre les inversions de courant que certaines charges inductives, comme des relais ou des moteurs, peuvent créer.

L'interface peut servir de contrôleur de moteur bidirectionnel. Pour mettre un moteur sous ou hors tension, il suffit de le placer entre le connecteur de sortie et celui de mise à la terre. Lorsque l'ordinateur émet un 1, la sortie est chargée et le moteur démarre. Une sortie zéro met le moteur hors tension. En connectant le moteur entre les deux sorties d'interface vous pouvez commander la direction du mouvement du moteur. Si l'ordinateur envoie deux sorties identiques (deux 0 ou deux 1), la même tension apparaîtra sur les deux sorties d'interface et aucun courant ne traversera le moteur. Un 1 sur l'une des sorties et un 0 sur l'autre donnera une différence de tension qui fera tourner le moteur dans une direction.



Construction de l'interface

Tout d'abord, vous devez construire le boîtier qui logera l'interface. Les dimensions de la carte correspondent exactement à celles spécifiées pour le projet de boîtier tampon. Le boîtier doit être percé afin de pouvoir recevoir les prises et la fiche bus (et la fiche à 12 voies si cela est nécessaire).

Dès que les prises sont fixées au boîtier, nous devons établir des connexions entre elles. En utilisant un bout de fil, connectez ensemble toutes les prises de mise à la terre (noires). Puis prenez un bout de câble-ruban à 9 voies d'une longueur de 12 cm, et reliez un brin au fil connectant les prises de mise à la terre. Reliez les huit autres brins, par groupes de deux, aux quatre prises de sortie (rouges).

Coupez maintenant la carte de montage à la bonne dimension (45 trous × 16 bandes) et enlevez la demi-rangée de trous à l'une des extrémités. Conservez cette chute puisqu'elle sera utilisée lors de la construction de l'autre interface. Faites maintenant les pistes comme indiqué dans la photographie A.

Soudez d'abord les composants passifs : la prise puce, la prise bus et les fils de liaison.

Les résistances sont alors soudées, puis les diodes. Assurez-vous que ces composants sont bien installés. Ensuite, soudez les huit transistors, aux bonnes positions. Soudez finalement les connexions aux prises comme dans la photographie B, et installez la puce. Vous pouvez maintenant installer la carte dans le boîtier, l'interface est alors prête à être utilisée.

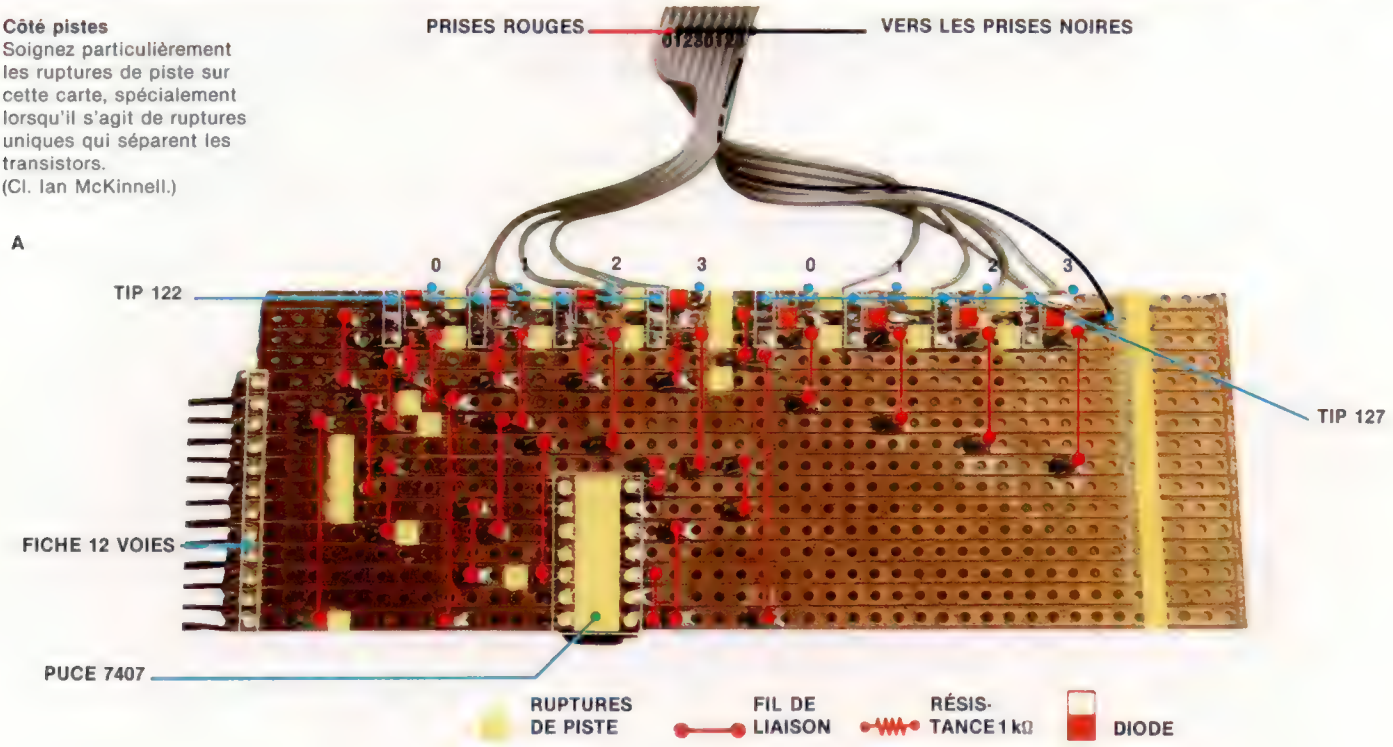
Liste des pièces

Quantité	Article	N° Maplin
1	Carte de montage de 50 trous par 24 bandes	FLO7H
4	Résistance 1 kΩ	M1K
8	Diode 1N4001	QL73Q
4	Transistor TIP 122	WQ73Q
4	Transistor TIP 127	WQ74R
1	7407	QX76H
1	Prise à 14 broches DIL	BL18U
4	Prise 4 mm rouge	HF73Q
4	Prise 4 mm noire	HF69A
4	Fiche 4 mm rouge	HF66W
4	Fiche 4 mm noire	HF62S
1	Prise 12 voies	YW19V
1	Boîtier 120 × 65 × 40 mm 2004	LH60Q
1	Prise à 12 voies	VW30H*

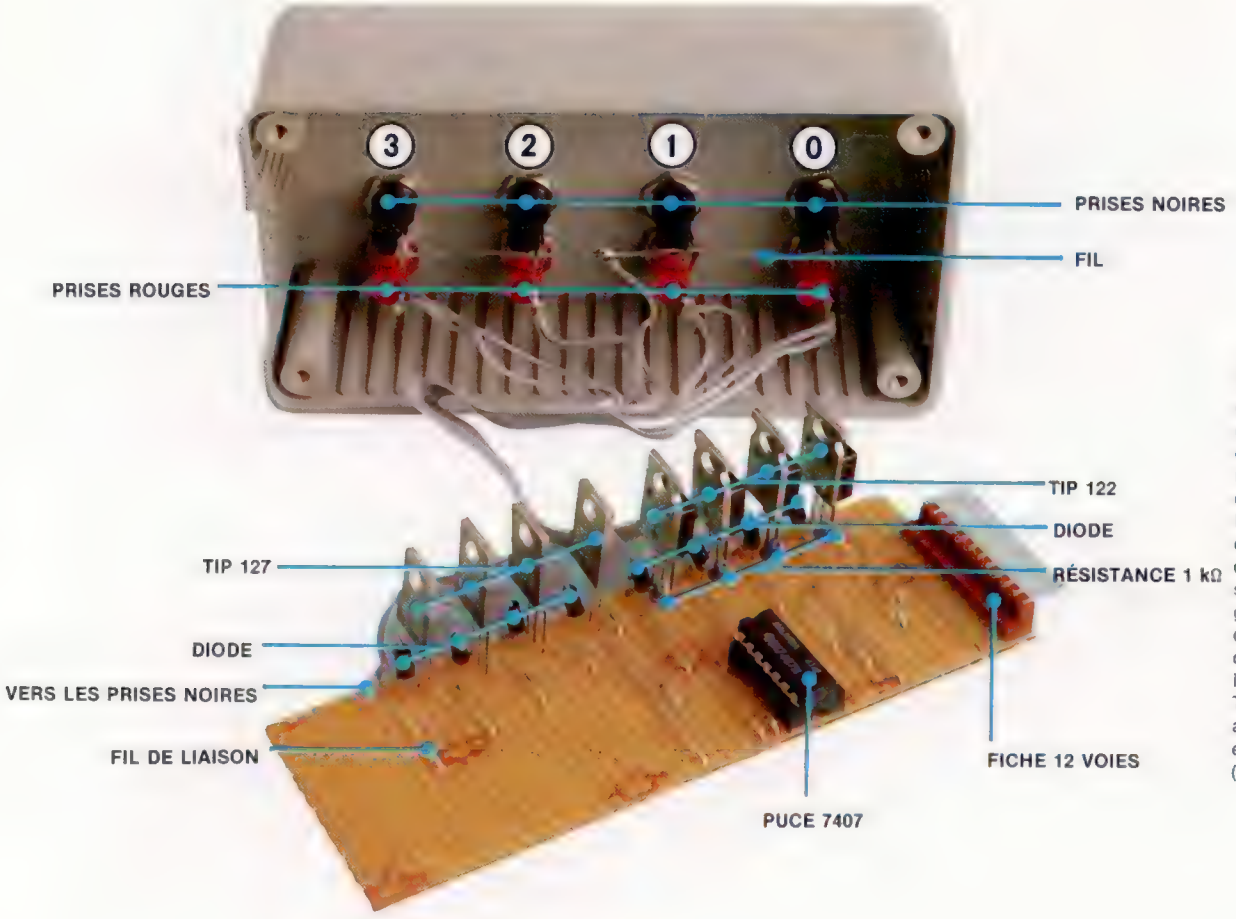
* Ce dernier élément est facultatif. Il prolonge le bus du système au-delà de cette interface afin de pouvoir brancher d'autres interfaces simultanément. Il devrait vous rester du fil et du câble-ruban provenant du projet précédent.

Côté pistes
Soignez particulièrement les ruptures de piste sur cette carte, spécialement lorsqu'il s'agit de ruptures uniques qui séparent les transistors.
(Cl. Ian McKinnell.)

A



B

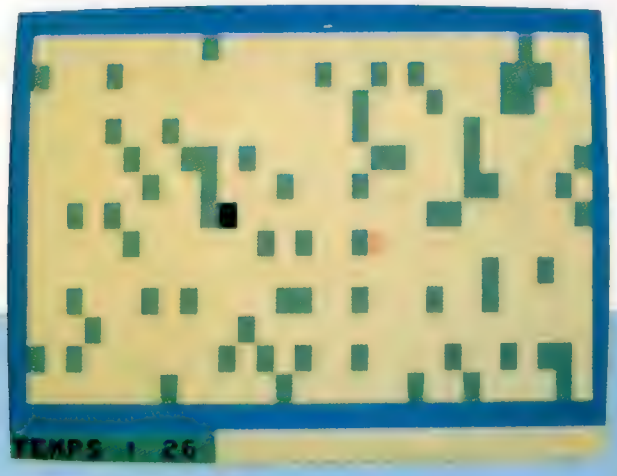


Côté des composants
Les TIP 122 sont situés du côté droit de la carte, et leurs quatre diodes sont connectées par l'extrémité noire à la piste latérale et par l'extrémité argent à la piste suivante. La ligne du câble-ruban provenant des prises noires est soudée à l'extrémité gauche de cette piste. Les quatre autres diodes sont connectées de façon inverse parmi les TIP 127. La fiche à 12 voies apparaît à l'une des extrémités de la carte.
(Cl. Ian McKinnell.)

La poursuite d'Alice

Pas question de vous essouffler dans de vaines poursuites d'un voleur. Pierre Monsaut a écrit ce programme en BASIC pour le micro-ordinateur Alice de chez Matra.

Le voleur s'est échappé, emportant le magot. Il se cache dans la ville, et vous avez trente minutes pour le débusquer et l'arrêter. Attention, pas de précipitation ! En effet, si vous vous jetez sur lui sans réfléchir, il y a toutes les chances pour qu'il vous file entre les doigts. La meilleure façon de vous y prendre est de l'aborder de côté. (Efficace à tous les coups, à condition de ne pas le rater !) Si vous ne vous sentez pas assez sûr de vous, attaquez-le de face, ce qui est plus facile mais moins efficace car moins discret. Encore un conseil : n'essayez pas de le poursuivre ; cela ne vous mènerait à rien, car il est aussi rapide que vous. Observez plutôt ses mouvements comme un vrai détective. Quand vous le verrez tourner en rond, approchez-vous sans faire de bruit et surgissez au bon moment. Mais rappelez-vous, le temps presse !



Déplacement :
Z : haut
Q : gauche
S : droite
W : bas

```
5 REM  **************
10 REM * POURSUITE *
15 REM **************
19 REM S=SCORE
20 S=0
25 REM V$=CARACTERE VOLEUR
30 V$=CHR$(128)
35 REM P$=CARACTERE JOUEUR
40 P$=CHR$(191)
50 GOSUB 2000
104 REM
105 REM BOUCLE PRINCIPALE
106 REM
109 REM MOUVEMENT JOUEUR
110 D$=INKEY$
120 D=(D$="Q")-(D$="S")+32*((D$=
"Z")-(D$="W"))
130 IF D<>0 THEN D0=D
135 REM DECOMPTE TEMPS
150 T=T-0.1
160 PRINT@ 480,"TEMPS :";INT(T+1
);
165 REM TEMPS ECOULE?
170 IF T<0 THEN 490
180 P=P+D0
190 C=PEEK(16384+P)
195 REM VOLEUR ATTRAPE?
200 IF C=128 THEN 4000
205 REM OBSTACLE?
210 IF C<>159 THEN P=P1
220 PRINT@ P1,CHR$(159);
230 PRINT@ P,P$;
240 P1=P
245 REM DEPLACEMENT VOLEUR
250 V=V+DV
255 REM OBSTACLE?
260 IF PEEK(16384+V)<>159 THEN G
OSUB 600
270 IF PEEK(16384+V)<>159 THEN 2
50
280 PRINT@ V1,CHR$(159);
290 PRINT@ V,V$;
300 V1=V
310 GOTO 110
484 REM
485 REM FIN
486 REM
490 D$=INKEY$
500 IF R<S THEN R=S
510 PRINT@ 166,"TEMPS ECOULE";
520 PRINT@ 234,"SCORE :";S;
530 PRINT@ 266,"RECORD :";R;
540 PRINT@ 326,"UNE AUTRE ?";
550 D$=INKEY$
560 IF D$="" THEN 550
570 IF D$<>"N" THEN 20
580 END
594 REM
595 REM OBSTACLE
596 REM
600 D2=D2+1
610 GOSUB 900
620 IF PEEK(16384+V1+DV)=159 THE
N V=V1+DV:RETURN
630 D2=D2-2
640 GOSUB 900
650 IF PEEK(16384+V1+DV)=159 THE
N V=V1+DV:RETURN
660 D2=D2-1
670 GOSUB 900
680 V=V1+DV
690 RETURN
900 IF D2>4 THEN D2=D2-4
910 IF D2<1 THEN D2=D2+4
920 DV=(D2=1)-(D2=3)+32*((D2=2)-
(D2=4))
930 RETURN
1994 REM
1995 REM INITIALISATION
1996 REM
2000 CLS 2
2005 REM TRACE DU CADRE
2020 FOR I=0 TO 31
2030 PRINT@ I,CHR$(175);
2040 PRINT@ 448+I,CHR$(175);
2050 NEXT I
2060 FOR I=1 TO 13
2070 PRINT@ I*32,CHR$(175);
2080 PRINT@ I*32+31,CHR$(175);
2090 NEXT I
2095 REM POSE DES OBSTACLES
2100 FOR I=1 TO 70
2110 GOSUB 3000
2130 PRINT@ P," ";
2140 NEXT I
2145 REM AFFICHAGE VOLEUR
2150 GOSUB 3000
2160 V=P
2170 PRINT@ V,V$;
2180 V1=V
2185 REM AFFICHAGE JOUEUR
2200 GOSUB 3000
2210 PRINT@ P,P$;
2220 P1=P
2230 T=30
2240 D0=0
2250 DV=0
2260 D2=0
2270 RETURN
2994 REM
2995 REM POSITION ALEATOIRE
2996 REM
3000 P=RND(414)+32
3005 REM POSITION OCCUPEE?
3010 IF PEEK(16384+P)<>159 THEN
3000
3020 RETURN
3994 REM
3995 REM GAGNE
3996 REM
4000 FOR I=1 TO 5
4005 REM SIRENE
4010 SOUND 35,10
4020 SOUND 5,10
4030 NEXT I
4040 S=S+1
4050 GOTO 50
```

Simple arithmétique

Examinons en détail plusieurs programmes en langage machine qui montrent comment de simples opérations mathématiques sont effectuées à l'aide du jeu d'instruction 6809.

A ce stade du cours, nous sommes capables d'assembler des instructions pour former un programme. Il nous faudra tout d'abord examiner de nouvelles instructions et de nouveaux modes de représentation des données. Commençons par un simple programme qui convertit un nombre décimal codé binaire (BCD) dans sa représentation binaire.

Un nombre décimal codé binaire est une façon de représenter un nombre décimal dans une forme binaire, qui est particulièrement utile, lorsqu'on a affaire à des processeurs 8 bits. Le nombre décimal 69, par exemple, est équivalent à la représentation BCD %01101001 : les 4 bits de gauche (0110) sont l'équivalent binaire de 6, et les 4 bits de droite (1001) égalent 9 décimal. Ainsi, en utilisant BCD, nous obtenons un équivalent décimal tout à fait différent de ce qu'on aurait en convertissant le nombre *binaire* %01101001 (qui équivaut à 105 décimal).

Notre programme de conversion nécessitera quelques nouvelles instructions ; considérons-les successivement :

• LSR (Logical Shift Right, « décalage logique à droite ») : chaque bit de l'opérande est décalé d'une case vers la droite. Le bit le plus à droite est reporté sur le bit de retenue du registre de code condition du processeur, et un zéro est placé dans le bit le plus à gauche de l'opérande.

• AND : c'est la fonction ET appliquée entre chaque bit d'un registre et le bit correspondant de l'opérande, le résultat étant placé dans le registre. Cette instruction est surtout utilisée pour *masquer* certains bits : si un registre contient 1 dans un bit, alors AND aura pour effet de copier ce second bit dans le registre ; si le bit de registre contient 0, le résultat sera toujours 0. Par exemple, l'effet de AND sur une valeur de registre de %00001111 avec un emplacement mémoire consiste à copier seulement les 4 bits de droite de l'emplacement dans le registre. Ainsi :

```
%00001111 valeur registre ;
%10110110 AND valeur emplacement mémoire ;
%00000110 résultat dans le registre.
```

• MUL : multiplie le contenu des registres A et B, le résultat étant dans le registre D (registre à 16 bits formé par la réunion de A et B). Il existe très peu d'autres processeurs 8 bits qui admettent la multiplication comme code opération (opc).

• SWI (SoftWare Interrupt, « interruption logiciel ») : c'est une manière commode de terminer un programme en langage machine, en rendant le contrôle au système d'exploitation. Nous examinerons plus en détail cette instruction lorsque nous considérerons le système d'interruption dans un autre fascicule. Voici le programme BCD-Binaire :

• Spécifier la valeur dans le compteur d'emplacement :

```
    ORG $1000
```

• Stocker 58 BCD en BCDNUM et réserver 1 octet en BINNUM :

```
BCDNUM FCB %01011000
BINNUM RMB 1
```

• Charger 58 BCD dans le registre A et masquer le chiffre inférieur. Le stocker en BINNUM :

```
START  LDA BCDNUM
       ANDA #%00001111
       STA BINNUM
```

• Charger 58 BCD dans l'accumulateur A et décaler le chiffre supérieur (les 4 bits de gauche) vers la droite :

```
SHIFT  LSRA
       LSRA
       LSRA
       LSRA
```

• Charger 10 (décimal) dans le registre B et multiplier par le contenu de A :

```
MULT   LDB #10
       MUL
```

• Le résultat est 16 bits dans le registre D, mais comme ce résultat ne peut être supérieur à 90 ($10 \times 9 = 90$), seul l'octet inférieur de D est nécessaire. L'octet inférieur est dans le registre B — donc additionner son contenu à BINNUM et stocker le résultat :

```
ADDIT  ADDB BINNUM
       STB  BINNUM
```

• Ainsi, nous avons le nombre BCD en BCDNUM et l'équivalent binaire stocké en BINNUM. Nous pouvons enfin retourner au système d'exploitation et terminer le code source :

```
RETURN SWI
END
```

Complément à deux

Les exemples de programmation que nous avons donnés jusqu'à présent impliquaient tous une arithmétique simple, et nous continuerons encore un peu dans cette veine.

Voyons d'abord le problème du *signe* — par lequel nous désignons les nombres positifs et négatifs. La méthode la plus commune pour représenter les nombres négatifs dans un empla-

Décimal	Binaire
−7	1001
−6	1010
−5	1011
−4	1100
−3	1101
−2	1110
−1	1111
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

cement mémoire d'ordinateur ou registre est la forme appelée « complément à deux ». Pour obtenir le complément à deux d'un nombre binaire, nous inversons tous les chiffres (changer tous les 0 en 1, et vice versa) et additionnons 1 au nombre ainsi obtenu. Ainsi, le complément à deux de 0101 est 1011.

Mais comment utiliser cela pour effectuer des opérations impliquant des nombres signés? Tout d'abord, considérons l'ensemble des nombres qui peuvent être représentés : un registre à 8 bits ne peut contenir que 256 configurations pouvant servir à représenter les nombres positifs compris entre 0 et 255, *ou* les nombres négatifs et positifs compris entre −128 et 127. (Un registre à 16 bits peut contenir des valeurs de 0 à 65535 *ou* de −32768 à 32767.) Nous donnons en marge une table montrant comment une représentation binaire est faite pour les valeurs décimales de −7 à 7.

En regardant le tableau, vous remarquerez que les nombres négatifs ont tous un 1 dans la position de bit la plus significative (à gauche). De même, tous les nombres positifs ont un zéro dans cette position.

Comme on peut le constater en examinant ce tableau à 4 bits, nous pouvons définir des propriétés de base pour les mathématiques signées, fondées sur le complément à deux :

• Le complément à deux d'un nombre négatif donne son équivalent positif, et vice versa.

• Le bit le plus significatif est toujours nul pour un nombre positif, et égal à 1 pour un nombre négatif. Cela permet de reconnaître si un nombre est positif ou négatif.

• Le complément à deux de zéro est nul (1111 plus 1).

• L'addition et la soustraction peuvent être effectuées comme d'habitude, et tout résultat aura le signe correct.

Vous voudrez peut-être essayer quelques additions et soustractions simples pour vérifier la légitimité de la dernière propriété. La multiplication est cependant plus difficile avec des nombres signés. L'instruction MUL, que nous avons utilisée dans le programme BCD-Binaire au début de cet article, traite les contenus des registres A et B comme des nombres sans signe. Si nous voulons multiplier deux nombres signés, nous devons le programmer nous-mêmes.

Quiconque a fait de la programmation comprendra que nous sommes extrêmement limités dans ce que nous pouvons faire en utilisant les simples programmes « linéaires » dont nous nous sommes servis jusqu'ici. La seule chose utile consiste à employer l'une des formes fondamentales de structure de contrôle :

• La *sélection,* par laquelle nous choisissons entre deux différentes voies d'action (telle l'instruction IF en BASIC).

• La *répétition* dans laquelle nous répétons une séquence d'opérations :

1) tant qu'une certaine condition reste vraie (structure WHILE...WEND);

2) jusqu'à ce qu'une certaine condition devienne vraie (REPEAT...UNTIL); ou bien

3) un certain nombre de fois (FOR...NEXT).

Toutes ces structures dépendent de la possibilité de tester une condition pour voir si elle est vraie ou fausse, le sort le plus commun de la condition étant de savoir si une variable a une certaine valeur ou non. En langage d'assemblage, nous avons besoin de toutes ces structures, et c'est pourquoi il nous faudra pouvoir tester les valeurs contenues dans les registres. En général, nous pouvons tester directement deux possibilités seulement (si une valeur est nulle ou non, et si elle est positive ou négative). Avec des instructions supplémentaires, il est néanmoins possible d'effectuer d'autres sortes de tests.

Ces conditions sont rendues disponibles par l'utilisation du registre de code condition (CC), que nous avons évoqué précédemment. C'est un registre à 8 bits, mais à l'encontre des registres du 6809, nous ne nous intéressons pas aux valeurs qui y sont stockées. Notre objet est plutôt l'état (1 ou 0) de chaque bit individuel. Cinq des huit bits sont consacrés à des conditions du type que nous avons discuté, les trois autres au maniement d'interruptions (que nous examinerons par la suite). L'un des cinq, H (Half, « drapeau de demi-retenue »), concerne presque uniquement l'arithmétique BCD, et ne nous intéresse pas pour le moment. Les quatre autres sont :

• C : le drapeau de retenue (Carry), qui contient le chiffre de retenue provenant du bit le plus significatif après une opération arithmétique. Il a également une fonction utile, lorsque nous voulons décaler de 1 bit le contenu d'un accumulateur; certaines opérations de décalage mettent le dernier bit dans C. Ce bit, par exemple, peut être utilisé pour tester si un nombre est pair ou impair en y mettant le bit le moins significatif pour le tester. C'est le bit 0 (le moins significatif) du CC.

• V : le drapeau de dépassement (oVerflow), mis à 1 chaque fois que le résultat d'une opération arithmétique est trop grand pour le registre qui est supposé le contenir. C'est le bit 1 du CC.

• Z : le drapeau de Zéro, mis à 1 lorsque le contenu d'un registre est nul. C'est le bit 2 du CC.

• N : le drapeau Négatif, qui est une copie du bit le plus significatif (bit de signe) d'un nombre dans un registre; autrement dit, il est mis à 1 si le nombre est négatif. C'est le bit 3 du CC.

L'un des aspects les plus difficiles de la programmation en langage d'assemblage est de garder trace de l'état des drapeaux. Toutes les instructions ne mettent pas les drapeaux, et certains sont mis selon le contenu de l'accumulateur, tandis que d'autres peuvent dépendre d'autres registres. La procédure la plus sûre consiste à tester seulement les valeurs dans un accumulateur, et ce, là où la valeur requise apparaît, puisqu'il est difficile de s'assurer que les drapeaux ne sont pas modifiés par l'intervention d'instructions.

Les drapeaux sont testés au moyen d'instructions de « branchement », qui équivalent à la commande BASIC GOTO. Le 6809 utilise des branchements relatifs (plutôt qu'absolus) presque exclusivement. La différence est qu'un branchement relatif transfère le contrôle d'une telle quantité d'octets en avant (ou en arrière), tandis qu'un branchement absolu transfère le contrôle à une adresse spécifiée. L'effet, cependant, est le même. Il fait une distinction entre branchement court, où la grandeur est exprimée dans un seul octet (– 128 à 127), et branchement long, qui peut aller n'importe où en mémoire.

Le 6809 a un grand nombre d'instructions de branchement, et nous les introduirons au fur et à mesure de nos besoins. Les exemples suivants illustrent les instructions utilisées pour tester et comparer les valeurs contenues dans les accumulateurs, et l'emploi d'instructions de branchement pour les procédures de choix et de répétition.

• ANDCC : il n'est pas possible de charger des valeurs directement dans le registre CC, mais il est bon de mettre tous les drapeaux à zéro avant de commencer à les utiliser. Le meilleur moyen pour cela est de se servir de l'instruction ANDCC, qui opère exactement comme une commande AND, en utilisant des zéros comme masques dans les positions de bit que nous voulons employer.

• SUB (SUBtract, « soustraire ») : l'opérande est soustrait de l'accumulateur, qui met les drapeaux C, V, Z et N dans le résultat (le drapeau H est également mis, si la soustraction est à 8 bits).

• CMP (CoMParer) : elle fonctionne exactement comme SUB, sauf que le contenu du registre est laissé inchangé. Comme dans SUB, les drapeaux C, V, Z, N (et peut-être H) sont mis.

• BRA (BRAnchement inconditionnel) : c'est exactement l'analogue de la commande BASIC GOTO.

• BGT (Branch if Greater Than zero, « branchement si supérieur à zéro ») : c'est un test pour les nombres signés. Le branchement a lieu si Z est nul (le nombre n'est pas zéro).

Pour permettre que le bit de signe soit mis incorrectement en cas de dépassement, soit N doit être nul et V aussi (simplement non négatif), soit N doit être à 1 et V aussi (négation incorrecte due au dépassement), d'autres tests similaires pour les nombres signés sont BGE, BLT et BLE.

• BLO (Branch if LOwer than zero, « branchement si inférieur à zéro ») : c'est un test non signé, puisque cela n'a pas de sens d'inspecter N avec des nombres non signés. Le branchement a lieu si le drapeau C est mis, indiquant une retenue dans une soustraction. Des tests similaires sont BLS, BHI et BHS.

• Un programme pour trouver le plus grand de deux nombres signés à 8 bits stockés en $3000 et $3001. Le plus grand des deux nombres doit être placé en $3002. D'abord, étiqueter les nombres :

```
NUM1   EQU   $3000
NUM2   EQU   $3001
ANS    EQU   $3002
       ORG   $1000
```

• Le code commence : les drapeaux CC sont mis à zéro et le premier nombre est chargé. Il est comparé à l'autre nombre :

```
       ANDCC #%11110000
       LDA   NUM1
       CMPA  NUM2
```

• Si NUM1 est le plus grand, alors le programme se branche en FINISH. Sinon, il charge le second nombre dans le registre A. Quel que soit le nombre dans le registre lorsque FINISH est atteint, il est stocké en ANS, et le programme revient au système d'exploitation et se termine (END) :

```
       BGT   FINISH
       LDA   NUM2
FINISH STA   ANS
       SWI
       END
```

Directives originales
Les différents effets que les directives d'assembleur et les instructions de langage d'assemblage ont sur le compteur d'emplacement de l'assembleur et sur les contenus de mémoire sont mis en évidence dans cet exemple.

Directives originales

```
ZONE      ZONE   ZONE              COMPTEUR      CONTENU
LABEL     OPC    OPÉRANDE          D'EMPLACEMENT DE MÉMOIRE
*---------DEMONSTRATION-------------
RESET     EQU    $F100             $0400         ???
INDEX     EQU    16                $0400         ???
MASK1     EQU    %01101010         $0400         ???
                                   $1000         ???
          ORG    $1000
                                   $1000         ???
CR        FCB    16                $1000         $10
                                   $1001         $7F
MEMTOP    FDB    $7FFF             $1002         $FF
TABLE1    RMB    7                 $1003         ???
                                   $1004         ???
                                   $1005         ???
                                   $1006         ???
                                   $1007         ???
                                   $1008         ???
                                   $1009         ???
ERRMSG    FCC    /ERROR/           $100A         $45
                                   $100B         $52
                                   $100C         $52
                                   $100D         $4F
                                   $100E         $52
          CLRA                     $100F         $4F
          END                      $100F         ???
*-------SYMBOL TABLE----------------
RESET  F100  INDEX  0010  MASK1  006A
CR     1000  MEMTOP 1001  TABLE1 1003
ERRMSG 100A
```

Aucun ORG n'a été émis, de sorte que l'adresse d'emplacement est celle de l'assembleur par défaut. L'adresse n'est pas affectée par EQU, et les contenus mémoire ne sont pas encore définis.

Ceci met l'emplacement comme spécifié, mais les contenus mémoire restent indéfinis.

FCB stocke l'opérande dans l'octet adressé par le compteur d'emplacement.

FDB initialise 2 octets.

RMB réserve 7 octets (contenus indéfinis), en incrémentant le compteur d'emplacement de ce nombre.

Les codes ASCII de la chaîne opérande sont placés en mémoire par la directive FCC.

Enfin, une opération de langage d'assemblage ! Il n'y a pas d'opérande : nous n'avons que 1 octet pour l'opc.

Directive qui n'affecte pas le compteur d'emplacement.

Voici comment les symboles seraient stockés dans une mémoire de travail pour son propre usage pendant l'assemblage.

Carte 6809 (fin)

Voici, avec l'aimable autorisation de la firme nord-américaine Motorola Inc, la seconde et dernière partie de la carte référence du programmeur 6809.

		Modes d'adressage																5	3	2	1	0
		Immédiat			Direct			Indexé			Étendu			Inhérent								
Instruction	Formes	Op	~	#	Op	~	#	Op	~	#	Op	~	#	Op	~	#	Description	H	N	Z	V	C
LSL	LSLA													48	2	1	A, B, M — c b7 b0	•	↕	↕	↕	↕
	LSLB													58	2	1		•	↕	↕	↕	↕
	LSL				08	6	2	68	6+	2+	78	7	3					•	↕	↕	↕	↕
LSR	LSRA													44	2	1	A, B, M — 0 b7 b0 c	•	0	↕	•	↕
	LSRB													54	2	1		•	0	↕	•	↕
	LSR				04	6	2	64	6+	2+	74		3					•	0	↕	•	↕
MUL														3D	11	1	A × B → D (sans signe)	•	•	↕	•	9
NEG	NEGA													40	2	1	$\overline{A}$ + 1 → A	8	↕	↕	↕	↕
	NEGB													50	2	1	$\overline{B}$ + 1 → B	8	↕	↕	↕	↕
	NEG				00	6	2	60	6+	2+	70	7	3				$\overline{M}$ + 1 → M	8	↕	↕	↕	↕
NOP														12	2	1	Pas d'opération	•	•	•	•	•
OR	ORA	8A	2	2	9A	4	2	AA	4+	2+	BA	5	3				A V M → A	•	↕	↕	0	•
	ORB	CA	2	2	DA	4	2	EA	+	2+	FA	5	3				B V M → B	•	↕	↕	0	•
	ORCC	1A	3	2													CC V IMM → CC				7	
PSH	PSHS	34	5+[4]	2													Entre Registres sur Pile S	•	•	•	•	•
	PSHU	36	5+[4]	2													Entre Registres sur Pile U	•	•	•	•	•
PUL	PULS	35	5+[4]	2													Sort Registres de Pile S	•	•	•	•	•
	PULU	37	5+[4]	2													Sort Registres de Pile U	•	•	•	•	•
ROL	ROLA													49	2	1	A, B, M — c b7 b0	•	↕	↕	↕	↕
	ROLB													59	2	1		•	↕	↕	↕	↕
	ROL				09	6	2	69	6+	2+	79	7	3					•	↕	↕	↕	↕
ROR	RORA													46	2	1	A, B, M — c b7 b0	•	↕	↕	•	↕
	RORB													56	2	1		•	↕	↕	(	↕
	ROR				06	6	2	66	6+	2+	76	7	3					•	↕	↕	•	↕
RTI														3B	6/15	1	Retour d'interruption					7
RTS														39	5	1	Retour de sous-programme	•	•	•	•	•
SBC	SBCA	82	2	2	92	4	2	A2	4+	2+	B2	5	3				A — M — C → A	8	↕	↕	↕	↕
	SBCB	C2	2	2	D2	4	2	E2	4+	2+	F2	5	3				B — M — C → B	8	↕	↕	↕	↕
SEX														1D	2	1	Signe étendu de B sur A	•	↕	↕	0	•
ST	STA				97	4	2	A7	4+	2+	B7	5	3				A → M	•	↕	↕	0	•
	STB				D7	4	2	E7	4+	2+	F7	5	3				B → M	•	↕	↕	0	•
	STD				DD	5	2	ED	5+	2+	FD	6	3				D → M:M + 1	•	↕	↕	0	•
	STS				10 DF	6	3	10 EF	6+	3+	10 FF	7	4				S → M:M + 1	•	↕	↕	0	•
	STU				DF	5	2	EF	5+	2+	FF	6	3				U → M:M + 1	•	↕	↕	0	•
	STX				9F	5	2	AF	5+	2+	BF	6	3				X → M:M + 1	•	↕	↕	0	•
	STY				10 9F	6	3	10 AF	6+	3+	10 BF	7	4				Y → M:M + 1	•	↕	↕	0	•
SUB	SUBA	80	2	2	90	4	2	A0	4+	2+	B0	5	3				A − M → A	8	↕	↕	↕	↕
	SUBB	C0	2	2	D0	4	2	E0	4+	2+	F0	5	3				B − M → B	8	↕	↕	↕	↕
	SUBD	83	4	3	93	6	2	A3	6+	2+	B3	7	3				D − M:M + 1 → D	•	↕	↕	↕	↕
SWI	SWI[6]													3F	19	1	Interruption logiciel 1	•	•	•	•	•
	SWI2[6]													10 3F	20	2	Interruption logiciel 2	•	•	•	•	•
	SWI3[6]													11 3F	20	1	Interruption logiciel 3	•	•	•	•	•
SYNC														13	≥4	1	Synchroniser pour interruption	•	•	•	•	•
TFR	R1, R2	1F	6	2													R1 → R2[2]	•	•	•	•	•
TST	TSTA													4D	2	1	Test A	•	↕	↕	0	•
	TSTB													5D	2	1	Test B	•	↕	↕	0	•
	TST				0D	6	2	6D	6+	2+	7D	7	3				Test M	•	↕	↕	0	•

Op	code opération (hexadécimal)
~	nombre de cycles MPU
#	nombre d'octets de programme
+	signe « plus » arithmétique
−	signe « moins » arithmétique
·	multiplier
M	complément de M
→	transférer dans
H	demi-retenue (venant du bit 3)
N	négatif (bit de signe)
Z	zéro (réinitialiser)
V	dépassement, complément à deux
C	retenue venant de l'ALU
↕	test ; mettre à 1 si vrai, à 0 sinon
•	non affecté
CC	registre de code condition
:	concaténation
∨	OU logique
∧	ET logique
⊻	OU exclusif

1. Cette colonne donne un cycle de base et compte d'octet. Pour obtenir le compte total, additionner les valeurs obtenues à partir de la table de MODE D'ADRESSAGE INDEXÉ en annexe F.
2. R1 et R2 peuvent être n'importe quel couple de registres 8 bits ou 16 bits. Les registres 8 bits sont : A, B, C, DP. Les registres 16 bits sont : X, Y, U, S, D, PC.
3. EA est l'adresse effective.
4. Les instructions PSH et PUL requièrent 5 cycles plus 1 cycle pour chaque octet entré ou sorti en pile.
5. 5(6) signifie : 5 cycles sans branchement, 6 cycles avec branchement.
6. SWI met les bits I et F à 1. SWI2 et SWI3 n'affectent pas I et F.
7. Les codes conditions sont mis à 1 comme résultat de l'instruction.
8. La valeur du drapeau de demi-retenue est indéfinie.
9. Cas particulier — retenue si B7 est mis.

Page manquante
(publicité)

Page manquante
(publicité)